Frontiers in Artificial Intelligence and Applications

DESIGN PROBLEMS, FRAMES AND INNOVATIVE SOLUTIONS

Martin Džbor

IOS Press

VISIT…

DESIGN PROBLEMS, FRAMES AND INNOVATIVE SOLUTIONS

Frontiers in Artificial Intelligence and Applications

Volume 203

Published in the subseries
Knowledge-Based Intelligent Engineering Systems
Editors: L.C. Jain and R.J. Howlett

Recently published in KBIES:

Vol. 196. F. Masulli, A. Micheli and A. Sperduti (Eds.), Computational Intelligence and Bioengineering – Essays in Memory of Antonina Starita
Vol. 193. B. Apolloni, S. Bassis and M. Marinaro (Eds.), New Directions in Neural Networks – 18th Italian Workshop on Neural Networks: WIRN 2008
Vol. 186. G. Lambert-Torres et al. (Eds.), Advances in Technological Applications of Logical and Intelligent Systems – Selected Papers from the Sixth Congress on Logic Applied to Technology
Vol. 180. M. Virvou and T. Nakamura (Eds.), Knowledge-Based Software Engineering – Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering
Vol. 170. J.D. Velásquez and V. Palade, Adaptive Web Sites – A Knowledge Extraction from Web Data Approach
Vol. 149. X.F. Zha and R.J. Howlett (Eds.), Integrated Intelligent Systems for Engineering Design
Vol. 132. K. Nakamatsu and J.M. Abe (Eds.), Advances in Logic Based Intelligent Systems – Selected Papers of LAPTEC 2005

Recently published in FAIA:

Vol. 202. S. Sandri, M. Sànchez-Marrè and U. Cortés (Eds.), Artificial Intelligence Research and Development – Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence
Vol. 201. J.E. Agudo et al. (Eds.), Techniques and Applications for Mobile Commerce – Proceedings of TAMoCo 2009
Vol. 200. V. Dimitrova et al. (Eds.), Artificial Intelligence in Education – Building Learning Systems that Care: From Knowledge Representation to Affective Modelling
Vol. 199. H. Fujita and V. Mařík (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the Eighth SoMeT_09
Vol. 198. R. Ferrario and A. Oltramari (Eds.), Formal Ontologies Meet Industry
Vol. 197. R. Hoekstra, Ontology Representation – Design Patterns and Ontologies that Make Sense

ISSN 0922-6389

Design Problems, Frames and Innovative Solutions

Martin Džbor
The Open University, Milton Keynes, MK7 6AA, United Kingdom

IOS Press
Amsterdam • Berlin • Tokyo • Washington, DC

ISBN 978-1-60750-067-4
Library of Congress Control Number: 2009937746
doi: 10.3233/978-1-60750-067-4-i

Publisher
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

Dedication

To my wife, my parents and my brother who were all very supportive at various stages of researching and writing this book.

This page intentionally left blank

Foreword

In this book, I present, illustrate and empirically validate a novel approach to modelling and explaining the nature of engineering design. The main outcome of this work is the formal definition of *problem framing* and the formulation of the Recursive Model of Framing in Design. The model (abbreviated as 'RFD model') represents a formalisation of a grey area in the science of design, and sees the design process as a recursive interaction of problem framing and problem solving.

The proposed RFD model is based upon a cognitive phenomenon known as *(reflective) solution talkback*. Previously, there were no formalisations of the knowledge-level interactions occurring within this complex reasoning operation. My RFD model is thus an attempt to express the existing knowledge in a formal and structured manner. The RFD model is applied to the knowledge-level description of the conducted experimental study that is annotated and analysed in the defined terminology. Eventually, several schemas implied by the model are identified, exemplified, and elaborated to reflect the empirical results.

In the book I propose a set of principled schemas on the conceptual (knowledge) level with an aim to make the interactive patterns of the design process explicit. These conceptual schemas are elicited from the rigorous experiments that utilised the structured and principled approach to recording the designers conceptual reasoning steps and decisions. They include

- the refinement of an explicit problem specification within a conceptual frame;
- the refinement of an explicit problem specification using a reframed reference; and
- the conceptual re-framing (i.e. the identification and articulation of new conceptual terms)

Since the conceptual schemas reflect the sequence of the 'typical' decisions the designer may make during the design process, there is no single, symbol-level method for the implementation of these conceptual patterns. Thus, when one decides to follow the abstract patterns and schemas, this RFD model alone can foster a principled design on the knowledge level. For the purpose of computer-based

support, these abstract schemas need to be turned into operational models and consequently suitable methods.

Some examples of how the RFD model helped me and my colleagues in designing novel user-focused environments for web and semantic web navigations are mentioned towards the end of the book.

Acknowledgements

Many people contributed to my professional growth and in various ways left their mark on my work and on this book. My first thanks goes to Zdeněk Zdráhal and John Domingue from Knowledge Media Institute of The Open University, who recognised research potential in my early and rather fuzzy ideas about modelling design. A number of other people provided comments on the models presented in this book at different stages of my research, including Marc Eisenstadt, Marek Hatala, Joanna Kwiat, Paul MacEachern, Paul Mulholland, Tammy Sumner, Victoria Uren, Michael Valášek, and Stuart Watt. I cannot forget my colleagues at the University of Technology in Košice, Slovakia (Tomáš Sabol and Ján Paralič), and Tokyo Metropolitan Institute of Technology, Japan (Shuichi Fukuda), with whom I started investigating partial aspects of designing, which eventually led me to forming this line of research.

I am also grateful to my examiners, Bashar Nuseibeh from The Open University and David Robertson from the University of Edinburgh for their helpful and encouraging feedback during the defense of my PhD thesis that forms the core of this book. The experimental work would not be possible with the enthusiasm of Petr Horáček and Lukáš Trejtnar from Czech Technical University, Prague.

My greatest debt is to my parents, my brother and my wife, whose support and encouragement became an endless source of confidence and motivation driving my research in the UK. The dedication of this work to them is only a little acknowledgement of all that support.

Finally, I want to thank to all the readers who decide to open this book and learn about my view of design and designing. I would also l like to invite you, the readers, to share your opinions, thoughts, and even formal models to respond to my claims.

Contents

Chapter 1

Executive Summary

In this book I present a perspective on the interpretation and solution of engineering design problems. Engineering design is a class of problems that are collectively referred to as "ill-structured" [Sim73]. Among the characteristics of this class, I emphasise one that appears as a thread throughout the book. Namely, a design solution includes the actual designed artefact or product, *as well as* the refined or extended specification of the problem to be solved. The explicit problem specification is not given, but evolves to respond to the development of our understanding of the design solutions (products). This is an important feature of all ill-structured problems – the problem can be better understood (that is, structured) only *while* developing and reflecting on its (partial) solutions.

A corollary of the view presented in this book is that the explicit specification of any non-trivial design problem is inherently incomplete at the beginning and throughout the design process. A set of statements about the desired states becomes the problem specification only when the designer accepts the produced design solution.

1.1. Class of problems

In this book I am interested in engineering design. I understand the design task as Chandrasekaran [BC85], Gero [QG96] or Goel [Goe97]. Accordingly, design can be defined using the following tenets (the essence of my original work is expressed in points 4 and 5):

1. a design problem can be specified as a set of requirements and constraints that refer to functions of the designed artefact

and/or properties of the design elements;
2. a design solution is developed in terms of structural elements and relations among them assuring the proposed structure meets certain desired functions and/or properties;
3. knowledge of a particular domain consists of simpler or more complex mappings from the set of structural elements to the functions and properties, through various relations;
4. mappings that form a domain theory depend on a particular design perspective (frame), i.e., a point of view from which the designer describes the conceptual world of designing;
5. we define a design perspective as a circumscribing frame imposed on the current design problem and created using familiar concepts from the design cases tackled in the past – this corresponds to Schön's observation that designers shape the problems by seeing them as more familiar situations [Sch83]

1.2. Problem solving strategies in design

Design, like other ill-structured problems, does not have a single, generally applicable strategy or problem solving method that would be effective for all cases. Nevertheless, it is possible to highlight a few typical strategies that are repeated in different design problems, and research into engineering design produced many models of the design process. However, the majority of models attends to the design process at the level of 'explicit knowledge'.

The main difference between my perspective and the existing ones can be flagged in this sentence: *The problem is given – these are the required features, and the task is to find a solution satisfying the requirements.* The initial premise of the statement is flawed; the problem is only 'given' in explicit terms at the end, when we know its solution (the designed artefact). Thus, any robust model of a design process needs to take into account the fact that design is about problem framing (formulating, articulating, or interpreting) that evolves alongside the construction of a solution (artefact, product).

In chapter 5 I present a model of the design process that satisfies this condition. The Recursive Model of Framing in Design (RFD) sees design as an interplay of two knowledge-level reasoning actions that are 'packaged' into two predicates – *specifies* and *satisfies*. The former attempts to define the explicit vocabulary to describe the design

problem sufficiently for the subsequent problem solving. This articulation is bound by so-called *conceptual design frame* that corresponds to a certain view at the problem at hand; it is an interpretation of the incompletely described situation. A frame serves as a pool of opportunities, from which various solutions addressing the particular specification may be constructed. The process of solution construction is not blind and random. The available concepts are chosen and put together so that the final structure satisfies the current explicit specification of the design problem that is interpreted in a particular design frame.

An interplay of the two predicates occurs when, for one reason or another, the artefact that satisfies the explicit specification of the problem is not accepted as a design solution. In other words, the chosen interpretative frame does not allow specifying the design problem to the full extent. Consequently, the problem may be re-interpreted or re-framed, and a new conceptual design frame may be articulated. In a modified frame, new conceptual building blocks become available; new products may be articulated and constructed to satisfy the amended design frame. A sketch of such interplay is in Figure 1.1.

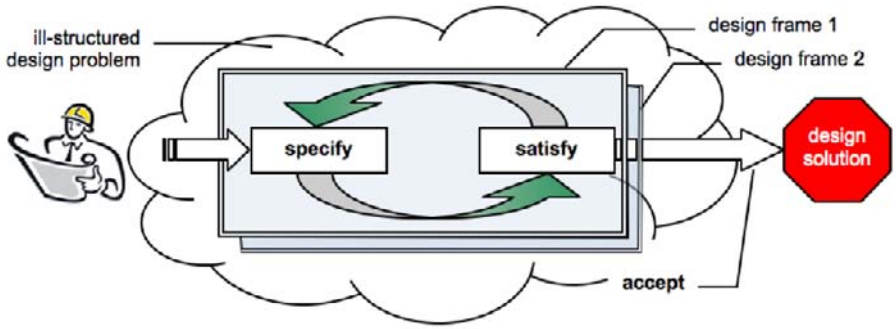


Figure 1.1. Interplay between problem specification and satisfaction within a particular design frame.

1.2.1. Level of explicit knowledge

Processes that can be modelled at the level of explicit design knowledge are associated with the second predicate (*satisfies* in Figure 1.1). This phase of the design process is well researched, and therefore it is

not the focus of my book. As an illustration, I mention some applicable models of how this knowledge-level action may work using the following reasoning strategies:

- *abduction from the domain theory* – assigns a set of structural elements to the desired functions (see assumption 1 in section 1.1;
- *deduction in the domain theory* – ensures the speculatively abduced set of structural elements is consistent with the domain theory by generating additional consequences for a particular element;
- *consistency maintenance* – ensures the solution developed within a particular domain is a consistent and valid selection of allowed structural elements

1.2.2. Level of (often) inarticulate processes

In addition to the explicit forms of reasoning, there is a portion of reasoning in design that is not expressed in the same, formal language. Not only it is not expressed formally, it is not even expressible explicitly. For instance, subjective assumptions, designer's intentions, 'feelings' and expectations influence the design process, although typically they cannot be explicitly articulated to the full extent. *Framing a design problem* is an example of this type of reasoning. The ability to interpret a problem, to frame a vaguely defined design situation, is one of the most valued design skills. Another process in creative and innovative design that resists an unambiguous explicit articulation, is the designer's reflection on the design decisions. As with framing, the ability to reflect "distinguishes design masters from the novices" [Sch83].

In my book, I focus on the operation of conceptual framing and see the *design frame* as a knowledge-level operation that aims to circumscribe incompletely defined design space. A design frame cannot be given; it must be constructed and *re-constructed* during design. This frame re-construction, or simply *re-framing*, is the core of a recursive frame development, and I present several schemas featuring various forms of re-framing in design.

Part I

Introduction to Design, Modelling and Framing

This page intentionally left blank

Chapter 2

Outline of the Model of Framing

This short chapter can be seen as a scenario for doing design as envisaged in my model. In section 2.1, I look at what decisions are typically made during the process of designing. The observations made in section 2.1 are expanded, refined, and explained later in chapters 3 to 5. Section 2.2 highlights the processes forming the subject of my enquiry. Again, a superficial sketch serves as an introduction to a complex research field, and further chapters work on the sketched subject more in depth.

2.1. Foreseen scenarios of design(ing)

Many models of design assume that the designer is given an explicit specification of the problem to which a solution is sought. According to [Cha90], such specification typically contains a list of desired functions, properties and various constraints on the product or manufacturing process. These models also assume that the designer works with a certain repository of components, modules and structures. The designer's task is to choose such a structure that meets the desired functionality.

Many techniques were developed to search for a design solution, based on different paradigms. Take two common techniques that are generally applicable in design – abduction from a logical theory, and logical deduction (see also section 5.3). First, the designer may abduce a sufficient structure that according to his or her domain theory has a potential to deliver the desired functionality. However, reasoning by abduction is not logically sound; the abduced structure can

be used only if it and its consequences are consistent with the logical theory, the requirements and constraints. The abduced structure must be evaluated against various explicitly formulated constraints, e.g., with respect to the compatibility, safety or manufacturing. By referring to constraints the designer seeks an answer to a question: "Are the given requirements and constraints met by the [abduced] structure (product, partial solution)?"

In addition to evaluating compliance with constraints, the designer must ensure that nothing inconsistent can be derived from the abduced choices. Therefore, it is necessary to deduce the consequences of believing a particular set of the abduced statements. Unlike abduction, reasoning by deduction infers only the necessary consequences that are implied by a set of axioms, decisions, etc. Both operations mentioned so far seek a solution that is plausible for the purpose of the given design problem – a design solution or product that satisfies the explicit specification of the problem.

Now, there may be several situations happening as a result of the abduction, deduction, and consistency maintenance. The most desirable one is that at least one consistent solution exists in the given domain theory. According to the explicit models of design, this is the desired state, and the design process may stop. Second, a less desirable, albeit very frequent, situation occurs if no consistent structure can be abduced from the domain theory. In other words, there is an explicit contradiction between two or more constraints interpreted in the particular domain theory. Any such contradiction needs to be removed to continue towards a consistent solution. To that extent, there are two paths for the conflict resolution:

1. current abduction is discarded, the designer backtracks to the last consistent state, and looks for *another candidate abduction*;
2. current abduction is discarded but one *cannot abduce any alternative* suggestion in the current domain theory

Point 1 is can be achieved by many algorithmic or heuristic techniques for *backtracking* [Bla99, CvB01, Dec92, SF94]. Alternatives to backtracking include, e.g., truth maintenance and contextual extensions in common logic [dK86a]. The situation in point 2 is common in the real-world design, and requires a different approach. For instance, TRIZ [Alt84] algorithm identifies the minimal contradicting pair of requirements, and resolves it by an application of less common

analogy.

In TRIZ the designer has to perceive the problem from a different, perhaps novel and innovative, perspective to overcome the inconsistency. In other words, conflict resolution does not lie inside the problem solving theory. By referring to the analogous problem, the designer brings in new concepts, axioms and assumptions into the logical theory. Eventually, the conflict between the requirements is resolved not by backtracking but by re-interpreting the conceptual perspective. Unlike monotonic backtracking, such re-interpretation is a non-monotonic operation. This is an important finding because it bridges the gap between the theoretical perspective shifts argued by Schön [Sch83] and the practical search for inventions of Altshuller.

Looking at a design problem from a different perspective, a designer may appreciate new features, new conceptual objects s/he was unaware of before. A designer learns more about the design problem by shaping (framing) and re-shaping (re-framing) it [Sch83]. A shift of perspective or *re-framing* triggered by an explicit contradiction is one of the challenges that is not explained in many existing design models. A bigger issue is that it is not always possible to articulate the conflict explicitly. In the simple situation (point 1 above), there was at least one consistent solution in the domain theory. Suppose the abduced structures are indeed logically consistent with the current domain theory, and there are no logical contradictions present. However, the designer *reflects on* (that is, looks at and analyses) the achieved state of the design, and declares that 'he is not happy with such a solution' or 'she does not like it'. Is such a situation possible?

I argue that such a situation is not only possible, but it is typical for the *ill-structured* problems (see section 1.1). The incompleteness and ambiguity of the ill-structured problems literally calls for an extension of the problem solving theories. The designer's tacit dissatisfaction with the current solution is caused by a conflict between his or her expectations and the explicit problem solution and specification. The declaration of unacceptability precedes the inarticulate need for a perspective shift. Similarly as with the explicit contradictions, the shift may lead to the extension of the logical theory, which by definition introduces non-monotonic reasoning to design. This is a tangled situation, difficult to describe – there are no explicit contradictions in the original domain theory, but in spite of this, there is a perception of insufficiency.

Look at this 'inarticulate lack of consistency' as a selection of an incorrect or incomplete conceptual perspective (*frame*). The explicated design frame that was used for the derivation of the unacceptable solution is not synchronised with all the implicit or tacit expectations the designer may have. To resolve this conflict, a synchronisation between the explicit and tacit is desired. Intuitively, perspective shift or *re-framing* of the design problem look promising. However, unlike the situation with an explicit conflict, a tacitly perceived contradiction must be formulated explicitly before any changes to the perspective may occur.

Frame shifting aims to improve the designer's understanding of the design problem and to refine its interpretation. The frame shift starts by an articulation of new or novel concepts from which a different explicit problem specification and solution may be constructed. The explicit formulation of new design assumptions, requirements or constraints, based on the tacit expectations, is actually an opportunity for the "exploration of the ill-structured design space" [SCD+90]. Exploring the opportunities of a given design problem by framing and re-framing, the designers refine their knowledge of the domain and of the problem, as well as the criteria for declaring something a design solution. This refinement is essential in non-trivial problems and fosters a co-evolution of the problem specification and the design solution [NCB97].

The two techniques for conflict resolution are different in spite of looking similar. The former one that deals with the explicit contradictions can be translated into solving the task of *finding an alternative solution* (point 1 above) or *solving a differently framed problem* (point 2 above). In other words, using a different analogy helps interpret the contradictory statements and the axioms of the problem solving theory differently. In the latter case, what the designer tackles is the problem of *what is an alternative interpretation of the current design state*. Instead of resolving the conflicts, s/he is trying to learn more about the problem, discover the opportunities of the design task, and find what can be done with the 'dead end'.

The points raised in my introduction hint why design is a worthwhile research topic. On one hand there is a quest for a solution *satisfying* the problem specification, and on the other one, a quest for the conceptual frame, in which the problem can be *explicitly specified*, and interpreted (and later satisfied).

2.2. Models of reasoning techniques in design

Reasoning techniques applicable to design include, for instance, frame articulation and modification, conflict identification, and design reformulation. These operations rely on knowledge sources that complement the explicit and formal domain theories. They may go beyond common logic and account for the earlier-mentioned hard-to-articulate, non-monotonic reasoning processes. Listed in Table 2.1 is a brief overview of different operations typically observed in design.

The table summarises both explicit and tacit operations from the previous section. These knowledge-level reasoning steps are listed in the order they appeared in the discussion in section 2.1. My recursive model of framing in design (RFD) addresses those reasoning steps with a grey background in Table 2.1. These steps and concepts are defined formally in chapter 5, and extended in chapter 9.

Operations in Table 2.1 can be indexed using a reference to the level of explicitness of the knowledge sources that are used. A type of knowledge source may inform the character of an operation, which may range from formal to vague, tacit and inarticulate. Different reasoning operations emphasise what knowledge roles are manipulated rather than formulating some prescriptive methods. As I show later, these functional definitions reveal several interesting similarities between the different operations.

First, there is a repeated reference to the previous knowledge (e.g., knowledge of past designs, knowledge transfer, etc.). The second striking feature is a presence of design perspective or context, called *conceptual frame* among the used knowledge sources. These two features are closely related in the design reasoning. Perception of similarity, familiar interpretation of a design problem, or transfer of previously acquired knowledge are examples of reasoning based on the designer's experience. Analogy plays a more important role in the design than the usually acknowledged component re-use [WP97]. Its influence is found throughout design and is particularly visible in the conceptual phase (framing of the design problem in the familiar conceptual terms).

A *conceptual design frame* is an essential concept of the my recursive model of framing in design(-ing), and more space is given to this theme in chapters 5 and 9. One can see the process of understanding a design problem (i.e., framing it) as what Simon calls "*extra effort*

Table 2.1. Operations on different knowledge sources in design.

Operation	Formality	Knowledge type	Description
framing	mixed	requirements, conceptual frame (ontology)	Interpretation of given requirement in terms of familiar situations, selection a suitable design perspective, frame
abduction	formal	functional requirements, domain theory (mappings), structural elements	Production of candidate solutions in terms of structural elements from given requirements and domain theory
deduction	formal	domain theory (axioms, mappings), candidate solutions, functions	Derivation of necessary consequences in terms of functions and behaviours for the assumed structural objects using the domain theory
consistency check	formal	domain theory (axioms, mappings), candidate solutions, constraints	Assurance that problem solving theory does not contain contradictions, e.g., introduced by the abductive reasoning
conflict identification	formal	domain theory, TMS, candidate solutions	Trace a contradiction to the violated axioms, constraints or assumptions
refinement of problem specification	mixed	design frame, requirements, constraints	Explicit articulation of a commitment to a requirement or constraint that was implicitly present but 'untold'
frame modification	informal	requirements, constraints, assumptions, solutions, past cases	Search for an alternative frame and an articulation of analogous matches between the current design and past design cases
problem reformulation	informal	design frame, requirements, constraints, assumptions	Transfer of relevant knowledge using the (modified) frame and introduction of additional requirements, constraints or assumptions

designers spend before being able to solve the problems" [Sim73]. This process would not be possible without reasoning based on similarity, familiarity, and analogy.

In the literature, different terminology is used to refer to that 'extra effort'. Some talk about *problem interpretation* [CRR+90], others use terms *immersion* and *act of insight by which certain design elements are chosen* [DD95, CE96]. Some present this issue as an *explication of tacit intentions* [NSH94], and some refer to this operation as *situation shaping* [Sch83]. Among them Schön emphasises that conceptual shaping (or framing) is a fully-fledged operation that acts as a pre-processor for the rest of the design. A pre-processor that, in

general, influences the understanding of what is the problem to tackle and the deciding how to construct a solution to it.

The topics discussed in this book are only the tip of a massive iceberg, however, it is my firm belief that my proposals can be used to further enhance research and thus benefit the design theory and design support. My intention is to define and shape the subject of enquiry for the science of design at multiple levels. First, a recursive model of framing in design is proposed as a reasoning strategy. Its main purpose is to describe different reasoning actions that underlie such a complex activity as design. Second, I investigate the deployment of the proposed model and the theory of framing for the representation of design problems. The ultimate goal is to propose a seed of a methodology for supporting the designers tackling non-trivial design problems. In the next section I will present some motivation examples and additional background.

The examples in section 2.3 show what part of the design process and the designer's reasoning is the focus of the book. Before going into details, boundaries are drawn for positioning my enquiry and for clarifying the used terminology. The terminology includes commonly used terms, for instance, 'design', 'knowledge', or 'modelling'. To avoid any ambiguity, section 2.4 clarifies their meaning with respect to my argument and design theory.

2.3. Motivational scenarios

Herbert Simon included design among so-called *ill-structured problems* [Sim73] and argued that most of the real-life problems belong to this category, which is in fact the reason why something like science of design is needed. Features that characterise ill-structured problems include incompleteness and vagueness of the initial problem specification; this, in turn, contributes to seeing design more as an art than science. In addition to the incomplete beginnings, there is the absence of a well-structured problem solving space and the absence of clear criteria for the determination of solution suitability. Let us look at these features of design and relate them to the real-world scenarios. In particular, I look at three *knowledge-level issues*: knowledge incompleteness, applicability of knowledge, and knowledge dynamics.

Despite Simon's observations about the structure of design situations being a few decades old, much ongoing research was concerned

with the application of various formal reasoning techniques to real-life problems. Unfortunately, only very few techniques and reasoning methods can be successfully transplanted from a well-structured mathematical world to an ill-structured reality. One of the main reasons is the necessity to make decisions with incomplete information at hand. In the real world there are many exceptions to the well-defined rules; the real world has plenty of typical and default, but also exceptional features, which are able to overturn common logic.

Nevertheless, the defaults do not pose the most difficult obstacle for understanding design. Whether a statement is typical or exceptional it can be explicitly formulated and expressed using a suitable language. A statement may be forgotten in the initial problem specification, and may need to be added there later. However, in addition to the facts that are untold, designers also use knowledge that cannot be expressed in the same (formal) language as the explicit design knowledge. The following real-world situations show what is meant by *knowledge that may be used but resists formalisation*. The examples are drawn from different domains to illustrate the commonality of this problem.

2.3.1. Can you tell how you do it?

Michael Polanyi was one of the first to draw attention to "*tacit knowledge*"indexknowledge!tacit[1]. He defined this 'hidden knowledge' as something that is inherently present and used when tackling a problem, though it may not be possible to express or explain it explicitly. Polanyi illustrated tacit knowledge by describing cyclists who are able to cycle on any bike they are given. Many different models can be developed by physics explain some aspects of cycling as a motion. These models include formal definitions of different forces acting on the cyclist, the bicycle, and the road. Nevertheless, any such physical models will be of little use to the beginner who wants to learn cycling.

Cycling also relies on our ability to stay upright on a bike, keep our balance, and not fall off the bike. The vast majority of people when asked to explain the principle of balancing to a beginner, will be *able to do it* and show it. However, they will usually find they are *unable to say how* exactly they turn the handlebars, how they

[1]Term *tacit knowledge* was introduced by Polanyi in his 1966 book 'The Tacit Dimension'; the cycling example is from [CB99].

measure the forces acting on them, what forces are relevant, or what strength they apply. Simply, they keep their balance and avoid falling without having to know the physics and its explicit models. Although they are not able to describe what they are doing, neither when nor how they react, they clearly must have this knowledge because it is a prerequisite for riding a bike.

Polanyi claims that what people are talking about, is an explicit dimension of their knowledge. People attempt to give some explicit explanation based on their knowledge of mechanics or physics, but this explicit statement is insufficient for a novice cyclist to practice cycling. This inexpressible knowledge, according to Polanyi, belongs to the tacit dimension. Cook and Brown emphasise that explicit and tacit knowledge exist in the different dimensions of the same problem space [CB99]. These are very distinctive forms of knowledge and neither is a variant of the other one. They complement each other, and typically, one type can be used to acquire the other one, but never transformed to the other type [CB99].

Thus, tacit knowledge of balancing is required to allow one to ride a bicycle. An explicit description of what is happening when one rides a bicycle may serve as an aid in acquiring the necessary tacit knowledge of staying upright. On the other hand, however precise the 'cycling theory' is, it does not guarantee a novice will actually be able to ride a bike without falling off. It is not a problem of the amount or quality of explicit knowledge one has about cycling. The explicit form by itself simply cannot do all the necessary work; other dimensions clearly play a role. One can therefore distinguish *knowledge* as an explicit category that can be *possessed* from tacit *knowing* [CB99]; i.e., knowledge that can be *practised*, knowledge as a competence [New82].

Design is in its nature similar to cycling. As cyclists use their tacit knowledge of staying upright to maintain their balance, designers have tacit expectations when assessing the suitability of a designed artefact. They use implicit assumptions to interpret the design problem or to consider a particular decision in the solution construction. It is obvious that the designed artefact has to be suitable in respect to the given requirements and constraints, such as colour, size, compatibility, etc. However, designers often go beyond the explicit assessment to develop a better impression of their work. A nice example of such a deeper form of assessment can be found in the history of shawl

weaving [Sch83]. Slovakian peasants produced unique shawls woven of yarns dipped in home-made dyes. The dyes have been prepared for centuries using knowledge that was passed from one generation to the next. The inherited knowledge included both the skill (of practising the handcraft) and knowledge (of additives). When the weavers were given modern aniline dyes (supposed to simplify their work), the beauty of the shawls was "spoiled and lost" [Sch83].

This happened not because the aniline dyes were poorer or unsuitable for shawl weaving. In fact, they were richer and cheaper, and could replace the natural dyes. Unfortunately, such a straightforward logic completely failed and Schön suggests a reason why this failure may have occurred: The actual design process was crudely disrupted by the ready-made artificial dyes, and the weavers were not able to recognise good or bad strategies because they lost some vital information from the dying phase. Something they commonly used when preparing hand-made dyes was left out by the explicit problem of choosing an artificial dye.

In other words, the weavers' rich understanding of a complex procedure of dye preparation was abruptly replaced by explicit knowledge that had a narrower scope of validity. The explicit knowledge might have been correct with regard to the colours, but that was only a part of the information the weavers needed. They clearly evaluated all intermediate stages tacitly during the entire weaving process. Such an evaluation may have included the dye shade, the time and temperature of its preparation, and perhaps other aspects difficult to describe by explicit, physical quantities. Since the practitioners' knowledge from the tacit dimension cannot be readily externalised [CB99], the design process was oversimplified by the focus on the explicit facets.

2.3.2. Rules are not enough

The next example comes from a different domain – medical diagnosis. One form of diagnosing a disorder called Hashimoto's lymphocytic thyroiditis (HLT) uses sonographic images of the patient's gland [STS+01]. Consider a few background facts: HLT is a diffusive inflammation of the thyroid tissue that results in an insufficient production of thyroid hormones. Since thyroid hormones regulate the body metabolism and have impact almost on all other organs, their low production adversely affects patient's health.

HLT diagnosis is usually based on the visual assessment of the images of thyroid area acquired by a sonograph (sound wave echoing). Medical students learn that HLT is manifested by a particular character of the texture captured in the scan. They are told to focus on the "*echogenicity of the tissue*" (ability to reflect the waves) and "*textural structure of the parenchyma*" (its regularity and roughness). The guidelines are simple and unambiguous; the reality is different. The scans of thyroids with a positive HLT exhibit a large variability and ambiguity. There is not a single, uniquely and explicitly definable region in the scan to look at. The structural parameters to be assessed are vague and do not have crisp values that would clearly prove or reject suspected HLT.

To illustrate this vaguely defined problem, four snapshots are presented in Figure 2.1 (from [SSS+01]). The dotted line is added to each image to simplify the identification of the "*significant region*" that needs to be investigated. One can see irregularity of the region shape and size for different patients. Normally, the shape and position of this region would have to be established by a medical practitioner from a raw sonographic snapshot. Also, if one considers the explicit rule "*Look at echogenicity and textural structure of the parenchyma in a significant region.*", the process is not clearer. Unless one possesses the necessary knowledge of what to look at, one can hardly use the explicit guideline to obtain a meaningful diagnosis. The explicit account of the diagnosing process lacks an important piece of the puzzle – a chunk of knowledge, which is hardly formalisable – a context-dependent feeling for thyroid texture medics develop during their numerous exposures to the similar images that enables them to recognise variable forms of HLT.

Once the significant region is determined, the properties of the texture are examined. The practitioner takes into account the region itself, its texture granularity, smoothness, randomness, and many other features of the parenchyma. An answer to the question what is a "significantly rough" or "irregularly delineated" texture, is not straightforward. The practitioners find it difficult to express this kind of knowledge but they admit it is vital for the correct diagnosis. Rather than having learned this knowledge in the explicit terms, they talk about developing a feel for the situation or acquiring some intangible ability to exercise their knowledge through their past experience. It is this complementary knowledge and its relationship to

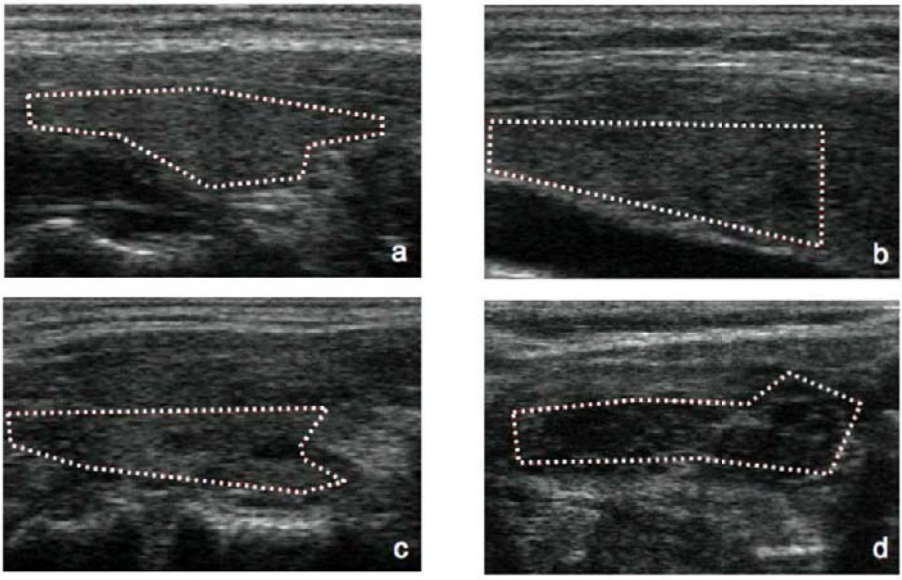


Figure 2.1. Healthy thyroid (a) and HLT-positive thyroids (b, c, d).

the explicit one that is the subject of this book.

2.4. Context of the book

This book is about design (in a broader sense) and about engineering design (in particular). I am exploring modelling the reasoning processes of a knowledgeable agent involved in designing non-trivial artefacts. The argument in the subsequent chapters weaves around three main themes addressing them from multiple perspectives:

- seeing *design* as a subject of enquiry that can be studied at different levels, including the *knowledge level*, and
- investigating *design* as a reasoning process exhibiting certain specific features and patterns that *can be modelled* at the level of agent's knowledge

Before analysing the design, let me commit to explicit interpretations of the three main concepts: *design*, *knowledge* and *modelling*. These basic concepts are commonly used in our everyday speech, and as such, they may have different meanings to different people. I offer my interpretations supported by the relevant literature, and

start with terms 'knowledge' and 'knowledge-level' in section 2.4.1. Section 2.4.2 looks at term 'design' and its characteristics. Finally, section 2.4.3 introduces the reader to the paradigm of 'modelling'.

2.4.1. Knowledge – a variable medium

We use term 'knowledge' and references to knowing very informal and interpret them rather freely. In Artificial Intelligence (AI), where also my modelling perspective fits, the community subscribes to the terminology coined by Allen Newell [New82]. He defined knowledge as "*a specific level on which information processing of an agent may be described. The new level was defined as an extension of the traditional hierarchy of information processing as it was well known in computer science* [NS76]. Newell positioned his knowledge level above the level at which physical symbols are manipulated to emphasise its more abstract nature.

Each level in the information processing hierarchy is described by the medium, by the basic components that can be assembled into systems, and by the set of laws governing the behaviour of such systems. Newell defines an agent solving a problem as the key 'knowledge system' with its main components being goals, actions, and bodies. The medium is knowledge, and the principle of rationality is the main law ruling this level – an agent's actions are determined by its goals. Knowledge level may be defined independently of the level below it (i.e., symbolic level), and has the following properties:

- knowledge is a kind of agent's *competence* enabling it to generate and carry out actions;
- knowledge is linked with rationality (i.e., it leads to *goal-oriented* reasoning and decision making);
- knowledge serves as a *specification of how* a symbol structure may achieve its purpose; and
- a body of knowledge on the knowledge level is *realisable by different* symbolic systems and representations.

Knowledge can be defined only functionally (i.e., what it enables) not structurally (i.e., in terms of unique physical objects). Thus, different symbolic structures (representations) may fulfill the function of the same knowledge – knowledge is not restricted to any passive medium, language or representation. *Knowledge* as a vanilla term coupling the agent's actions and decisions together is different

from the *represented knowledge*, which is an embodiment in some explicit representational framework. Following examples present different 'types' of knowledge with an aim to show what variability exists at the knowledge level.

Assume we describe the case of diagnosing HLT from section 2.3.2 at the knowledge level by ascribing a knowledgeable agent (e.g., a medical practitioner) some knowledge that would account for his actions when solving this particular, non-trivial problem. We can start with the knowledge of medicine, which includes human anatomy, clinical disorders, etc. Such a knowledge may tell us that thyroid is a gland in humans and that it produces a thyroid hormone. Our agent may have deeper knowledge about how this hormone affects the metabolism of a human body. Undoubtedly, such knowledge exists in the medical literature, and as such, is taught at any medical school. A vast portion of this knowledge could be expressed formally, for instance, using the following semi-formal statements:

- `type-of ( Thyroid, Gland )`
- `has-property ( Thyroid,`
  `property-value ( Shape, Butterfly-like ))`
- `affects ( Thyroid-hormone, Metabolism, Positively )`
- `IF greater-than ( content ( Thyroid-hormone ), 5.5)`
  `THEN activity ( Thyroid, Low )`
- etc.

In addition to this encyclopaedic knowledge that can be expressed in any existing formalism, there is some knowledge that will resist formalization as shown above. This is not to say that one language or an expressive framework is stronger than another. It may only be easier to express certain knowledge in a certain manner: e.g., instead of describing the position of thyroid using formal expressions, one can draw on sketches. Something as shown in Figure 2.2 provides a succinct account of the relative position of the gland against other organs. Those other organs may be easier to locate, they may be more visible, or simply they show lower variability than thyroid. It is thus reasonable to ascribe our agent also such an informal knowledge.

From section 2.3.2 we know that HLT is recognised using sonographic scans, and is exhibited through changes in the texture of the parenchyma. With a smaller or greater effort we could express the respective rules explicitly and formally, for instance, as `IF ... THEN`

... clauses. However, as shown in Figure 2.1, the scans show a high degree of variability in tissue texture. Such variations in texture would be hardly expressible in rules.

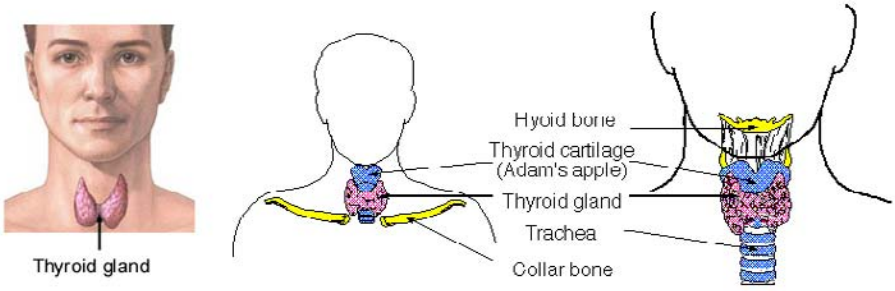


Figure 2.2. Informally represented knowledge of the thyroid position.

Therefore, one can ascribe to our agent also some knowledge that allows him to interpret roughness in the thyroid-specific context. This knowledge will be restricted to a particular method of investigation (sonograph), body part (thyroid), and the disorder (HLT). Similar images may be used for diagnosing other organs (e.g., X-ray scans of lungs) but in the context of a different organ, different properties may come to the fore. Nevertheless, the correct interpretation of roughness and granularity is an essential part of our agent's knowledge, and it may be impossible to express specific, measurable characteristics or formal reasoning chains for this knowledge type. One may say our agent relies on some tacit, intangible, and experiential knowledge.

All above items contribute to a knowledge-level model for HLT diagnosis. However, the existence of a model does not mean that the agent actually possesses all that vast amount of knowledge [New82]. Our model would be sufficiently complete if it could explain how the observed agents approach the given problem. Simultaneously, such a model should also arrive to similar conclusions as our agent.

Assume that during the tests we observe some agents perform extremely well; they reach their decisions with a high precision and confidence. Sticking to our model, they may have a better inference engine or may have additional, experience-based knowledge that gives them advantage. However, due to the variability of scans and the steady level of precisio these two suggestions lose their edge. If our knowledge-level model of the agent cannot explain the exceptional

behaviour, the model itself lacks an important bit that was overlooked before. We forgot to identify some knowledge used by the agent...

An example of such a hidden knowledge may be, e.g., the fact that HLT is an inflammatory disease, and as such can be proven by a presence or absence of antibodies in the patient's blood. Thus, in addition to the non-invasive sonographic scans, one may learn that invasive needle biopsy is an alternative way to ascertain HLT precisely and quickly. This is a straightforward knowledge described by a formal procedure of taking a blood sample and conducting a chemical analysis. Although being formal and straightforward, it is an 'unknown', as it was not captured in our initial model.

So far, we distinguished between knowledge valid in a specific context and knowledge that is general, applicable to many contexts. Another category for differentiating types of knowledge we mentioned is whether we are able to express it in some existing representation (e.g., predicate calculus or rules). In contrast, we also identified experience-based knowledge that is much harder to be represented explicitly. Another type considers the level of formalism chosen for representing a particular knowledge (e.g., production rules were more formal than sketches). Sketches, in particular, are a widely used mechanism for representing knowledge, particularly popular among engineers and designers [Goe95]. Yet another category considers whether knowledge is actually expressed explicitly in the current problem-solving task. In some situations, a problem solving agent may possess a particular knowledge but from various reasons it remains inactive – not expressed and represented; the agent behaves as if he was not aware of this knowledge.

Table 2.2. Classification of knowledge sources used in a typical problem situation

Knowledge category	**Applicable adjectives/characteristics**
Contextual dependency	context-free, contextual
Explicit representability	explicitly representable, tacit
Explicit presence	explicitly represented, implicit, hidden, forgotten
Level of formalism	formal, informal

The categorization is also depicted in Table 2.2; the first column corresponds to different dimensions or types of knowledge. Terms listed in the second column are the extreme ends of spectrum rather then crisp categories. In addition to the multiple dimensions of knowl-

edge as a medium, there are certain correlations among the adjectives taken from the different dimensions, as suggested in my review.

2.4.2. Design – a subject of enquiry

There are many definitions of what design actually is – almost as many as there are design researchers and publications about design. Nevertheless, it is possible to gather different views in a broad and a narrow definitions. For the former, the opinion of Simon [Sim69] and Smithers et al. [SCD+90] is helpful:

> Design is the concern of all occupations and professions that are engaged in converting the actual into preferred situations.
>
> Design task generally occurs when an external agent determines to change the status of its surrounding world. The main purpose of design as an activity is to deliver some outline how the desired changes can be accomplished.

According to the broad definition, design is a goal-driven, problem solving activity that subsumes other specialised activities such as re-engineering or configuration. However, it is possible to narrow the definition to express only the features that were long associated with the traditional design occupations such as architecture, engineering, or more recently computer programming, using Schön's words [Sch83]:

> A designer makes things. Sometimes he makes the final product, more often a representation, a plan, program or image of an artefact [...] He works in particular situations, uses particular materials, and employs a distinctive medium and language. Typically, this process is complex.

These definitions of design are domain-independent, and apply to town planning, technological processes or devices. Chandrasekaran [Cha90] breaks the definition into the essential aspects that suit particularly well the engineering problems:

- a set of *functions* to be delivered by an artefact (incl. those explicitly mentioned and those implied by the problem domain);
- a set of *constraints* to be satisfied by the designed artefact;
- a technology, that is, a repertoire of *components* available and relations among them

According to this view, the constraints may address design parameters, process of designing or process of manufacturing. A design

solution is formulated in a form of used components, modules, and relationships among them that ensure the satisfaction of the desired functionality. A set of components is a solution if it satisfies all the explicit and domain-specific (implicit) constraints. However, design is often under-specified [Goe94]; the desired functions or applicable constraints are incomplete. In some cases, in addition to the incomplete goals, the repertoire of available means is open-ended too. An artefact may be designed from the existing elements, or new elements may be recursively designed to deliver sub-functions.

The open-ended nature of design tasks prompted an intuitive categorization of design problems. Well known are well-structured (trivial) designs distinguished from the ill-structured (open-ended) ones [Goe94, Sim73]. Others promote the view whereby design may be categorised as one of the following [Ger90, Goe97]:

- *routine* – sets of design functions, components, variables and their values are fixed,
- *innovative* – essential functions and components are fixed but there is space for introducing something new into design, or
- *creative* – novel design components, variables, or values go beyond the existing framework, and significantly extend it.

I propose an extension to the above views and define design as a goal-oriented process leading from the *initial problem specification* toward an *acceptable design solution*. The specification includes the desired functions, properties, and constraints; whereas the solution is seen in a broader sense. A design solution includes the *refined problem specification*, the set of components achieving this amended specification, and the process leading to the formulation of the two. The design process glues together the evolving specification of design goals with the evolving specification of means (components, solutions).

2.4.3. Modelling – a paradigm for enquiry

One meaning of term 'model', according to Merriam-Webster dictionary, is "a copy" or "an image". This coincides with the general systems theory [Kli85], where term "modelling relation" is used isomorphically with term "similarity relation". Two systems are similar if they exhibit some commonalties and can be converted to each other

by a suitable transformation. For instance, system X may be an original, system Y may be another system similar to the original in some aspect and may act as its substitute in respect to certain features. System Y is called *modelling system*, and together with a relevant transformation makes up a model of the original system X.

Modelling is a relation of equivalence, and as such exhibits all properties of equivalent relations (including symmetry, transitivity and reflexivity). Whether or not a particular system is a model is usually a subjective, context-dependent decision of the creator and/or user of the model. If system Y can substitute the original, and has some advantages over it (e.g., is cheaper, less dangerous, or easier to compute), then it may well be a model of the original system. In other words, a model requires an original, and it is obvious that one original may have several different models that differ in some aspect (e.g., complexity or precision).

Models may be developed at different levels, including the symbol and knowledge levels (see section 2.4.1). The usefulness of modelling as an approach is illustrated in the following example: If one wants to design a structure made from reinforced beams for a multi-storey building, such a structure must be robust and safe to support the walls and must withstand winds of a given strength. Instead of constructing such a structure randomly and then checking whether it satisfies given constraints, the practitioner uses a model of it. The behaviour of the structure is calculated using a *mathematical model* rather then measured on the actual structure. Differential and algebraic equations act as a replacement of some behavioural aspects of the beam. The equations tell whether the structure complies with the constraints without expensively building the actual construction and they are much cheaper, safer, and faster to work with.

Another well-known type of models are scale models, whereby a real physical object (e.g., a body of a car) is replaced by its scaled counterpart. Instead of testing, e.g., aerodynamics on the original object, engineers obtain data from the scaled model. The scaled model has similar characteristics as the original in respect to the aerodynamics but it may completely fail to account for other features (such as driver's safety during a crash). Thus, for different purposes, it may be necessary to construct another type of model.

Models *are not the simplified versions* of the original systems; models must be equivalent to the original system, albeit only in a

sub-set of functions or behaviours. It may be easier or faster to work with a model than with the original, but this does not mean that a model is a simplified version of the original. The purpose of modelling is to develop systems acting similarly as originals and can more transparently scrutinized. In my book a model of design is developed that accounts for certain decisions of a human designer in the conceptual phase of the design, but does simplify the design in any way.

2.4.4. Epistemology of knowledge and knowing

Some authors distinguish the *possession* of knowledge from the *practice*. Both aspects of knowledge are important – to solve problems, one needs to *have some knowledge*, as well as *use it* in a knowledge-based action. The essence of this epistemological difference is captured in [CB99]:

> An accomplished engineer may possess a great deal of (...) knowledge; but there are plenty of people who possess such knowledge yet do not excel as engineers. (...) [I]f we want to understand the essentials of what accomplished engineers *know*, we need to look at *what they do* as well as *what they possess*.

From this argument originates also my perception of design, as defined in section 2.4.2, which treats it is an evolving interplay of problem specification and solution. In my perspective, design is an enquiry that draws on the extensive knowledge possessed by design practitioners. At the same time, it is an outcome of that knowledge applied in action (i.e., evolving process of design-*ing*) rather than the sum of the possessed knowledge.

This is also the reasons why I do not attend to the issue of representing knowledge. First, knowledge representation was a popular subject for several decades [Cha93, Ger90, GR91, ML84]. Second, I believe the utility of knowledge is in its presence in a particular (problem-solving) action, and not in its possession and representation. Although my view on the phenomenon of knowledge is opportunistic, it is aligned with the definition of knowledge as "*a competence-like notion being a potential for generating actions*" [New82]. In this context, a competence is an ability to have and simultaneously *to exercise*.

I conclude this discussion with an expressive sketch of the above-mentioned interplay. In Figure 2.3 knowledge possessed by an agent is categorised in a grid, in line with [CB99]. Each quadrant may be used

in an action as a means for the interaction with the design problem. Action gives the possessed knowledge its utility, and vice-versa, the possessed knowledge informs the agent's actions. The influence is reciprocal not only between possessed and exercised knowledge but also among the different types of possessed knowledge. However, my 'interplay diagram' is a more precise model of the behaviour of the different knowledge types. The right-hand sketch in Figure 2.3 models the reciprocal influences, as well as the generation of one knowledge type using another. Such a generation occurs in an action backed and justified by the agent's knowledge; the reciprocity is depicted by curved boldface arrows and the generation through an action by outward pointing straight arrows.

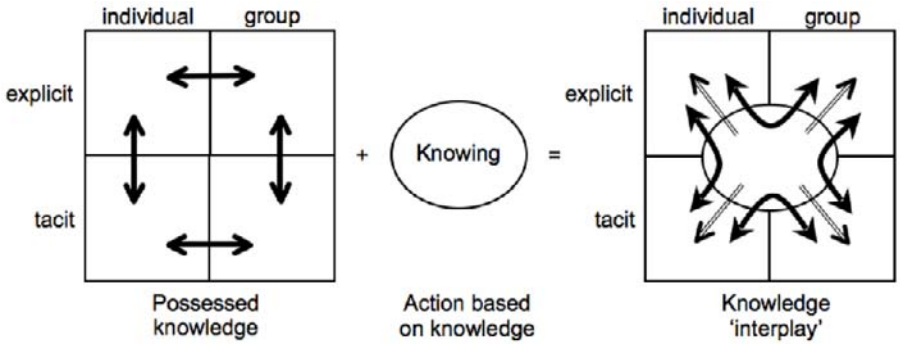


Figure 2.3. Knowledge and action interplay (adapted from [CB99]).

The implications of the duality of possession and exercise mean that a design problem cannot be articulated a-priori in any conceptual frame, unless the designer *applies* the frame and interacts with the artefacts allowed by that frame. The need to change the frame is not caused by the lack of possessed knowledge. On the contrary, it is triggered by this knowledge applied in action to find a way out from the deadlock.

The paradigmatic shift from the transformation to the *generation of knowledge* differentiates my approach from others. Using the terminology from sections 2.3.1 and 2.3.2, the explicit knowledge of the diagnostic rules, the artificial dyes or keeping balance while cycling does not replace the tacit knowledge of a practitioner. The tacit experience of the acceptability that one has to develop for the roughness of texture or for the dye composition cannot be simply explicated.

One can only apply the tacit knowledge in an action, and acquire (i.e., generate) some explicit account to rationalize that action.

My model of designing spins around recursive framing. The recursive model of framing explains how explicit design strategies may be related to the tacit ones. In particular, I focus on the issues of re-structuring and re-framing of the design problem one works on. I touch on several factors that enable us to see design as an interaction of knowledge-level problem specification and problem solution. Perhaps the most important one is that any satisfaction or specification makes sense only within a particular conceptual frame. Thus, instead of solving a design problem I talk about *satisfying an explicit specification of the problem that was made within a certain frame.*

If an acceptable solution cannot be found to a given problem specification, it points to an incomplete explicit account of the design problem. The problem must be re-interpreted, and I propose several techniques modelling this iterative refinement of the designer's understanding of the problem. One form of refinement is conceptual *re-framing*, when the axiomatic apparatus of a logical theory is amended by the introduction of new concepts and/or relations. These new concepts are then used as building blocks in finding a new candidate solution satisfying the re-framed problem.

I will show that it is possible to allow modifications of both the explicit problem specification and the solution. Both amendments may occur within a single design frame, or in a sequence of frames. From the epistemological point, re-framing may be viewed as stepping above the design problem and looking at it from a different angle. Such a step-back or 'creative leap' is associated with non-trivial design problems, and some, e.g., Schön, base the whole art of designing on the designer's ability to move between the different frames [Sch83].

2.4.5. Design and incompleteness

Since I referred to incomplete problem specification, it is useful to look at some approaches to tackle incompleteness, vagueness, and typicalities; in particular, the non-monotonic reasoning mechanisms, such as default logic [Rei80] or circumscription [McC80]. These approaches augment the predicate calculus so that it can handle the real-world scenarios with defaults and uncertainties. A reasoning system may be perceived as non-monotonic if a newly introduced axiom may in-

validate the existing theorems [MD80]. In other words, if a new observation is made in a particular world and one uses a non-monotonic logical system to reason about the properties of such a world, one is bound to re-validate all the inferences constructed in the incomplete theory, which did not account for the new observation. Some of the previous inferences may remain valid in the extended theory, others may lose their validity – i.e., non-monotonicity may change one's beliefs...

These augmentations are helpful if the explicit knowledge is incomplete and there exists a complete knowledge base, from which one can extract the missing information (e.g., a default value of a parameter). An important feature missing in these approaches is the capability of the logical systems to generate a new perspective by 'creating' new knowledge. And by 'knowledge creation' I mean more than merely retrieving a fact from a complete knowledge base!

One popular technique for coping with incompleteness is circumscription – a theory able to conjecture the conclusions along the following lines: "If we had had this particular information available then we would have been able to deduce such and such (conjectured) conclusion." This ability to conjecture something hypothetical is very powerful; however, since the circumscription deals only with the substitutions, how can one bring a particular information that would trigger the conjecture to the attention of the circumscribing operator? Consider the following situations as a sketch of a possible framework for further reasoning:

1. A logical theory is complete, and one only has to go through all predicates that exist in the theory and choose the 'right' one(s) as a conjecture;
2. One gives up logic as the framework for reasoning, and refers to the human being or another 'force majeure' that performs such a conjecture and returns the missing predicate;
3. Since the problem is not soluble in the current logical theory, one steps outside the logic and uses other knowledge sources that may not be expressible in the language of the current logical theory.

Alternative 1 corresponds to an ideal state; it assume the existence of "a super-human knowing everything" [Tom94]. Clearly, if this were the case, there would not be any need to worry about incompleteness.

Any task would be only locally incomplete with regard to the explicit knowledge an average designer has at any given moment. The design task would be a task of searching the knowledge base and explicating the relevant statements to extend the incomplete (explicit) design problem specification. Alternative 2 goes to the opposite extreme. It gives up the explicit theory and refers to an unquestioned authority. By referring to the higher authority it is possible to make decisions, but not to explain or justify them. In this alternative one sees why artistry or talent may come up in the context of designing. The decision of a higher authority is intractable to an external observer; the decision is to be accepted as an axiom, not to be proven, modelled or justified at the knowledge level.

First two situations do not give an answer to why and how the current perspective may be shifted. The third one offers a compromise: There may exist knowledge that cannot be encoded in a particular (logical) theory, because it makes statements about the theory itself [G̈30]. Such additional knowledge corresponds to various hard-to-articulate experiences and skills, which makes this situation a popular subject of investigation in many disciplines.

The third alternative assumes an existence of 'meta-knowledge' that could be deployed to manipulate the explicit (problem solving) levels of knowledge, and to shift the conceptual frame of the design problems. This meta-knowledge can also help to generate new information upon which future conjectures can be constructed. What is the form of this knowledge type and how can it be modelled at the knowledge level? Consider the following refinements of alternative 3:

3. . . .one steps outside the logic and uses other knowledge sources that may be available. . .

   (a) Higher-order knowledge is itself a logical theory; in addition to expressing statements about the objects and relations, it may express rules about when and how a particular axiom (conceptual object) can be amended.
   (b) Higher-order knowledge comprises generalised statements developed for a particular class of problems; these represent typical strategies, heuristics or best practices for the navigation in an incomplete design space.
   (c) Finally, this knowledge may be surfaced and critically applied using a reference to a large (not necessarily complete)

base of previous, familiar problems; e.g., through reasoning by analogy or similarity, case adaptation or generalisation.

The approaches outlined above differ in their dynamics. Knowledge in scenario 3a is most static; it is too prescriptive to account for flexibility and creativity in design. If, for example, it includes rules for choosing a fix when a particular violation occurs, then such a form is typically associated with routine, well-defined design tasks, where possible violations and fixes are known a-priori. An example of such a problem is the design of an elevator in the Sisyphus-2 domain [Yos92], where this higher-level knowledge is referred to as design fixes, and, in general, most of the constraints articulated for the elevator design have an associated fix. The fixes are applied deterministically when a particular constraint is violated by a partial solution.

Scenario 3b seems to be a more realistic approximation of the flexibility of the design processes. This scenario is represented, for instance, by the work on design prototypes [Ger90], where a prototype represents a class of products, contains its most significant features, relationships, etc. A prototype is not a fully-fledged, off-the-shelf solution; typically, it is a skeleton for further development and refinement. It is a point in the design space, from which further exploration may begin. Prototypes and heuristics are important for design practice, as they have potential to introduce new variables, parameters or concepts into the problem solving theory. Choosing a particular heuristic or prototype may focus the designer's attention on a specific constraint, which in turn, may lead to new conjectures.

Design prototypes, design heuristics or best practices are one class of strategies for *conjecturing* a further step. Another knowledge source with a potential for discovering new conjectures for a particular problem are specific design cases tackled in the past. The work aligned with scenario 3c covers case-based reasoning [RS89], case-based design [WP97], and reasoning by analogy [FFG89]. These approaches exhibit the greatest flexibility and dynamics as well as learning capabilities – all features that are desirable and connected with innovation and novelty. In this sense, one may understand the process of extending the case repository as a specific form of learning from a direct experience.

A repository typically contains design cases, as the designers understood them in the past. Therefore, one can claim that in addition to the individual designer's explicit knowledge such a repository may

also contain group knowledge of an organisation, and thus, it may reflect the designers' tacit, hard-to-articulate knowledge. Reasoning by analogy enables the designers to re-use the previously applied knowledge in different contexts. Designers may take different perspectives to understand the previous case and its relationship to the current problem. The previous design cases may be instrumental in interpreting (framing) the current problem in a usual or an unusual way.

2.5. Caveat on positioning of this book

Although I argue for the recognition of the influence of tacit and group knowledge on design, I do not attempt to analyse engineering design from the position of social psychology or ethnography. My work was not conducted from the perspective of human-computer interaction (HCI), which would perhaps be more focused on the actual interplay between the designer and the computational tools, their intuitiveness or just-in-time action [Eze00, MC02]. However, many methods used in HCI or ethnography can be used as input to my work. For instance, to validate the theory of iterative framing in design I followed the existing field studies performed by the ethnographers [RSP95]. Also, my analysis was carried out in accordance with the principles of the participatory design [ACM93, Mil93, Mul02]. This methodological inspiration helped to verify my theory of reflective and iterative design featuring (re-)framing.

Yet, my main emphasis remains on the reasoning strategies the designers use when tackling a vaguely defined problem, and on modelling the dependencies among these strategies. My ultimate objective is to better understand the nature of the conceptual phase of non-trivial engineering design projects. This objective informs also the preparation and conduct of my design studies, whose aim is to confirm or refute the assumption based upon the review of the literature that the designers frame their problems and reflect on their understanding of the problem. In addition, my studies show how the designers work on two simultaneous fronts – generation of a design solution and a further development of problem specification.

After setting up the stage and defining the essential terms and the background, let us move to the review of several perspectives on design and designing in the following two chapters.

Chapter 3

Perspectives on Design

In this chapter, I analyse design from a number of perspectives that ground the assumptions of my recursive model of framing in design. The structure of design problems is typically loose in respect to the reasoning strategies that may be deployed in the construction of a problem-solving space and efficient navigation within it. Different authors pay attention to different types of reasoning; a concise summary of such reasoning techniques applicable to design is, for instance, in [Cha90].

Section 3.1 covers the views on 'designing' from the artificial intelligence, knowledge modelling, technological, and cognitive perspectives. Section 3.2 is devoted to the introduction of two paradigms underlying my proposal: the concepts of 'reflection' and 'shaping the design situation'. My work directly extends the research on reflection in design. In section 3.2 I show that there is an intimate relation between the incompleteness and vagueness of the real world and the reflective (re-)shaping of one's understanding of a design situation. Section 3.4 concludes with basic features associated with design problems regardless of their complexity. This 'feature list' is my interpretation of the implications of the work reviewed in section 3.1, and it brings to the reader's attention the gaps in the science of design, which are likely to be an interesting subject for future investigation.

3.1. Design in multiple perspectives

Chandrasekaran [Cha90], Simon [Sim69], Coyne et al. [CRR+90], and others, approach design as a problem of searching through large

spaces of applicable (typically structural) elements. On the other hand, Oxman [Oxm90], Smithers et al. [SCD+90], Maher [GdSGM96], and others, emphasise that design is more an exploration of a problem solving space rather than any kind of search. This view is supported by the observation that in addition to searching, design is also about the construction of the problem space from the initial requirements. These are two basic views from a problem solving perspective, and are discussed in section 3.1.1.

When distinguishing methods used in the different phases of a design process, there are multiple definitions of what actually constitutes design. By emphasizing the operational aspects, design could be decomposed into a process of consecutive operations of lower or higher granularity [CRR+90, Ull97]. This coarse-grained sequence of operations like analysis, synthesis and evaluation may form rather complex chains, loops and hierarchies, as discussed in section 3.1.2.

Many consider design as a cognitive act covering a sequence of certain 'primitive' cognitive operations, whose main purpose is to strengthen a designer's insight into the problem [CE96, Cro97, Ger90]. In section 3.1.3, I look at terms like divergence, transformation and convergence, and approaches drawing upon cognitive science are reviewed; e.g., design patterns and prototypes [Ger90], or memory-based re-use [GdSGM96, WP97]. A specific cognitive phenomenon is creativity and innovation in design [Alt84, Ger90, TL99]. Since creativity is often cited as the typical feature of non-trivial, real-world designs, I present two contradictory views on creativity, and discuss the implications they have on modelling design formulation, solution discovery, and design support in section 3.1.4.

3.1.1. Search vs. exploration in design

Chandrasekaran [Cha90] defines design as a problem of search in a large space for objects that satisfy multiple constraints (see also section 2.4.2). Only a small number of objects in such a space constitute satisfactory solutions, and an even smaller number the optimal solutions. He argues that what is actually needed in design is knowledge to radically shrink the search space or to accelerate the navigation through it. Such knowledge could be expressed as a mapping between the primitive components (structures) and the states the designer wishes to achieve or avoid. Desirable design states could be expressed

as *desired functions* of the designed artefact [BC85]. Chandrasekaran later proposed a taxonomy of relationships between the (available) structures and (desired) functions [Cha93]. The idea of well-defined 'function–structure' links is re-visited in [IFVC93, QG96], and also in my RFD model (see chapter 5).

In general, these views capture causality between the primitive structures and the deliverable functions. They assume that the desired functions are a part of the problem specifications, and since the chains between the functions and structures are well defined in a particular domain, they may accelerate the search. Formally, there are two search strategies: (i) *backward chaining* or abduction when looking for a structure satisfying a function, and (ii) *forward chaining* or deduction when estimating the functionality of a particular structure. Term 'design' is reserved for the former searching strategy; i.e., how to find a structure that delivers the desired functionality. The latter strategy is known as qualitative modelling or prediction of behaviours from structure. This distinction splits two main schools of thought: One group deploys the causal chains between structures and functions to analyse "how things work and what they do" [GI91]; i.e., what functionality is implied by any given structure. Whereas another group emphasises the synthesis and looks at "how things are intended to work" [IFVC93]; i.e., how to design things that satisfy the designer's intentions regarding functionality. Design is then an activity covering the synthesis of an artefact, and analysis only plays a role in the validation and justification of the designed artefacts.

One can also meet the argument that analysis and synthesis are closely related, and the design comes out of their interplay. Accordingly, design cannot be reduced into a deployment of the existing structural links. Designers play an active role in design by not only using the information at their disposal, but also "actively manipulating it utilising a higher level knowledge and advanced control mechanisms" [Oxm90]. The distinction between analysis and synthesis is then fundamentally meaningless [LS92], and thus, the formulation of a design problem at any stage is not final. As the design progresses, new knowledge about applicable structures emerges. New knowledge may help to reveal inconsistencies in the existing problem specification, and eventually lead to a new understanding of the problem.

Smithers et al. compare the designers to the "adventurers or explorers who just landed at a new island and have very limited knowl-

edge of its geography" [SCD+90]. These may assume some features of the island, but to make a valid geographical report, they eventually need to go out, and explore the island. Every exploratory journey may describe a different feature of the island, and may bring new data to shape their original expectations.

The main argument of the 'design-as-exploration' school is that *a mere search is passive*; only the existing knowledge is manipulated and nothing new is uncovered (or discovered). On the contrary, exploration is about *the activity* and the development of knowledge. Search relies on heuristics to move through vast design spaces quickly [Cha90], but the heuristics need some a-priori knowledge. When such knowledge is not available, the search engines are unable to acquire it on the fly (in this sense, it is passive, too). Exploration, on the other hand, does not need a-priori knowledge. Existing knowledge is constantly re-shaped and re-validated as the design progresses and new solutions or relations become apparent.

Seeing design as an exploration seems to be more realistic from the epistemological perspective, too. Cook and Brown [CB99] but also Schön [Sch83, Sch95] report on the importance of (human) activity in designing artefacts and learning. They formally distinguish *knowledge* as something that is possessed from *knowing* as something that is actively practised. Knowledge is inherently passive and acquires its potential only when applied in an action; only then something new can be produced or generated out of the possessed resources. Search involves only the navigation in the possessed knowledge; whereas exploration involves an action, in which possessed knowledge is validated and refined.

3.1.2. Operationalisation of design process

Designers from an engineering background also attempt to split the complex design process into smaller chunks – primitive design operations or processes. Three major operations typically performed during the design of technological artefacts are [Asi62]: *analysis*, *synthesis* and *evaluation*. In analysis, the designer attempts to understand the problem and explicitly specify design goals. These goals are used in the subsequent stage, synthesis, where solutions are sought to satisfy the analysed goals. Finally, during the evaluation, the validity of synthesised solutions is assessed, the optimal alternatives are selected and implications are drawn about refining the goals. A cycle

emerges when the evaluation yields a need to re-examine the goals and the problem analysis. Repeated analysis consequently impacts the current solutions that may need to be re-engineered. This model of a design process is known as 'waterfall' (see Figure 3.1), as it assumes that the activity at any stage begins only when the previous stage was fully accomplished. However, when one looks at the details of the individual phases, the assumption of clearly divided stages is rather idealistic.

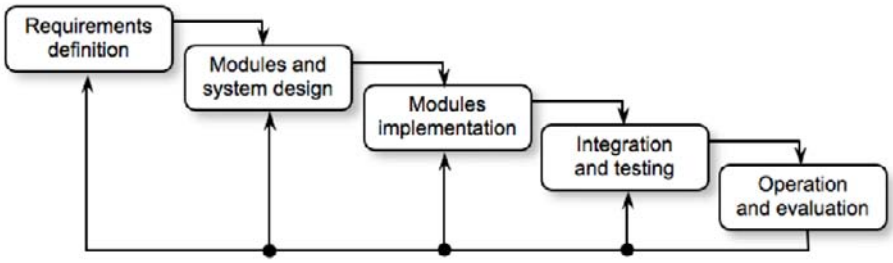


Figure 3.1. A sample representation of the waterfall model of design.

Because of the rigid distinction of the individual operations in the waterfall models, some authors suggested more sophisticated breakdowns distinguishing the following sequence of operations usually observable in the designs of technological artefacts [Ull97, NCB97]:

1. Planning
2. Development of engineering specifications
3. Development of conceptual solutions
4. Implementation of solutions and product refinement
5. Evaluation of the implemented product(s)

Unlike the waterfall model with its single loop, there are multiple feedbacks in the detailed models. A designer may interrupt the flow of the design process and return, for example, to the planning phase, or, after proposing a conceptual solution, the original specification may be amended to reduce the number of conceptual solutions. The problem with the operational views is similar to the dichotomy between the search and exploration: While search assumes a clear borderline between the analysis and synthesis, the exploratory paradigm is more modest about such clear separations. Another issue arises when attempting to order the distinctive operations with regard to

their causality or temporality. In general, there is only a small group of design problems, to which operational models fully apply. The vast majority of problems are less transparent, and rather than a sequence of primitive operations, their mutual interplay is idiomatic.

The operational view brings in several benefits, especially in an organisational context, where it underpins documentation and monitoring processes. However, a strict distinction of the design phases may compromise innovation and novelty in general. A lot of current design practice has "over-the-wall character" [Ull97]: Professionals involved in a product design work independently and communicate with each other only via a limited channel. As if they worked inside a walled yard and all their communication with the external world consisted of occasional scraps of information thrown over the wall.

Isolation is not a typical environment for humans to live and work in. The issue of isolation was investigated by psychologists [Red88], whose conclusions coincide with those of design theorists: Whilst some operationalisation is a useful approach to a complex issue, there remains a danger of re-inventing the wheel [Fis92]. When an unnatural division of activities is imposed on designers, as most of the waterfall operational models do, these models tend to oversimplify the real design process. Such oversimplification may lead the designers to committing the same failures repeatedly, and produce outcomes that have little to do with the real needs of the customers.

There are many attempts to improve the performance of the waterfall models; one being a spiral model of an entire product lifecycle. However, the spiralling models still separate operations that are typically intertwined. To rectify this the"Twin Peaks" [Nus01] model 'spirals' between two dual information spaces typical for software design – the model intertwines the requirements with the architectures to incrementally deliver the product. This model is one of few formal attempts ascribing equal importance to the development of requirements (i.e., problem analysis) and the development of solutions (i.e., product synthesis). This "inevitable intertwining of problem specifications and solutions" [SB82] is at the heart of the modern science of design as well as my book.

3.1.3. Cognitive aspects of design

From the cognitive viewpoint, the following cognitive processes are observable in the design process: Designers typically start with *a di-*

vergent strategy, whose task is to prepare a number of alternative paths; then they go through *a transformation phase* linking the prepared elements in a usual or in a novel way. Finally, *they converge*, work out the details, and implement them in a physical form. With respect to divergent thinking, user studies repeatedly report on the need to keep several possible design channels open in parallel [CE96]. This need is particularly strong in innovative and creative designs.

However, there is no single pattern of generating design ideas; on the contrary, multiple perspectives often broaden the knowledge a designer may have about a particular domain. The initial divergence may look distracting, but novel ideas do not usually appear suddenly – a designer or an inventor must be prepared for them. The divergent approach to the problem enables the understanding of the existing approaches in similar and different contexts. It helps the designers to discover interesting analogies with designs in other areas, and ultimately transform their own design space as well as their approach to a given design problem.

Several user studies give evidence of a causality between the divergent thinking at the beginning, the ability to transform the existing artefacts into novel ideas, and the convergence elaborating a transformation into a final product [Cro97, Cro99]. Often, a novel idea comes to life during the transformation and convergence phases, and its introduction may seem unexpected and abrupt. However, after a careful analysis, even an innovative concept can be traced back to the early phases of design through several transformations [Cro97]. Consequently, novelty has roots in less usual transformations of more familiar approaches.

This sudden appearance of an idea is known as *a creative leap*; i.e., it is not random [Cro97]. The essence of a creative leap is to construct a bridge from the familiar solutions to the unusual ones. An important assumption for a successful creative leap is that a designer has to be firmly standing among the familiar approaches. Just as jumpers do not jump without a previous warm-up, the designers need the 'warm-up' consisting of several transformations that make them aware of analogies and similarities [Cro97]. A typical example of a transformation is a combination of known approaches, as e.g., the example of designing a bicycle rack in [Cro97], in which the designers first propose 'a panel' as a model of a rack. An alternative proposal considers a rack as a kind of 'bag'. Both suggested alternatives have

their strengths and weaknesses, but their combination leads to something that is "as sturdy as a panel but also had side walls like a bag" [Cro97]. Figure 3.2 shows a schema of such a combination. An idea that sprung from a combination of "a bag on a panel" later evolves to a new concept of 'a tray'. This novel concept exhibits the desirable features of the common alternatives, and is a compromise between their negative features.

Figure 3.2. Combination of design ideas into a novel concept (based on [Cro97]).

Another cognitive phenomenon observable in the design is the emergence of new design features. A feature is understood as emergent when it comes unintended with the rest of the explicit (design) solution; often it requires a higher-level perception and imagination [Ger96]. Emergence usually manifests itself in unexpected features exhibited by a designed product. Although it may play an important role in triggering the novelty, the emergent features may be also harmful, as, for example, in a fast-activating airbag, where the activation speed is a desirable feature designed to protect a driver from the injuries after the impact. However, in certain cases, the fast-expanding airbag may inflict injuries worse than the impact itself. It may break one's neck or cause a brain concussion, in particular to children. These are not the features anyone would require or desire. These (negative) features emerged with a designer's decision to opt for a particular implementation or for a range of design parameters of the airbag.

I mentioned several times *familiar approaches* and *common solutions* that can be transformed into innovative ones. What is the source of these familiar solutions? One source, observed in the context of architectural design, are design prototypes [Ger90]. "A typical feature of architectural design is the existence of certain patterns that describe the essential attributes of the buildings but leave a freedom

for innovation and/or extension" [Ger90]. Design patterns and prototypes belong among memory-based design techniques [GdSGM96], together with case-based design (CBD) [WP97].

A typical application of memory-based techniques begins with articulation of a desirable pattern that addresses the design problem;e.g., "we want to design an office building" [GdSGM96]. The next step typically retrieves known prototypes or previous design cases that comply with the articulated pattern. The familiar approaches have different granularity, applicability or similarity, and one or more of them are then adapted to the current problem. The adaptation may involve the refinement of an applicable prototype (e.g., an office building with a rectangular geometry, open-space layout and space area larger than $400m^2$) or it may involve a combination of familiar solutions (as in the case of a bicycle rack).

The memory-based approaches to design differ in how the individual cognitive activities are represented and modelled, and in the source of familiar solutions. Designers may use a repository of past design cases in the CBD paradigm, or may use generalised prototypes or design models in prototype-based paradigm. Several types of models are useful in design [Ger90]: *A stereotype* is a model that can be copied without change, e.g., in the off-the-shelf or mass production. *Archetypes* are first or singular instances of a particular type/model; e.g., Rolls-Royce Phantom is an archetypal luxurious car. Finally, *prototypes* are generalised representations of the groupings of certain elements from similar design cases (e.g., a prototypic office building).

The familiar sources can be applied rigidly or flexibly. Here, several classes of design problems can be distinguished according to the flexibility of knowledge re-use [Ger90, Cha90]. Many design problems are *routine* or *well defined*, with a complete and strictly structured design space, all necessary parameters, values or means of deployment known in advance. In such a case, design is about the parameter instantiation. Another category involves *innovative problems*. Innovation takes place when a familiar structure acquires a novel mode of deployment; i.e., the design moves out of the well-defined tracks and the results are less predictable than with routine problems. Innovative designs use known parameters but assign them novel values.

The final category consists of *creative problems* with minor or major extensions to the familiar, well-defined design spaces. Typically, such extensions cannot be achieved by a novel deployment of known

structures and parameters. They require an articulation of new structures, relationships or paradigms, and products resulting from such creative extensions are often referred to as inventions [Alt84].

3.1.4. Is creativity predictable?

Terms creativity and invention widely used and abused in many books. There are two schools of thought explaining the creativity. The first one points out that creativity in design is more a social, issue when the ultimate judge of the product creativity is the society using it [WP97, Ste94]. Another group considers creativity to be a result of certain problem solving techniques [Alt84, Liu00, TL99].

Can a creative design process be repeated and still produce creative results? If a creative process is re-usable, are its modifications also considered creative? Is it the deployment of a product in a particular domain that may be judged innovative or creative? A particular solution may be known in other domains, but is it the unusual deployment that justifies the adjective 'creative'?

Creativity may be associated with both the product of any design method, as well as with the specific design procedure. The viewpoint seeing the product and its deployment as a measure of creativity is supported in the prototype-based [Ger90] or case-based design [GdSGM96, PG98], where a sufficiently large repository of past design cases (products) may trigger unexpected analogies or innovative explanations. Selected aspects of the case-based paradigm compiled in [WP97] demonstrate the close relationship between the creativity and the past design cases. Several features of CBD systems are relevant to creativity. CBD approach provides the designers with insights into how components were combined in the previous cases to deliver the desired performance. Reminders of the previous instances of design may thus improve the current ones by enhancing the creativity or imagination.

A familiar representation may have also an adverse effect on the designer's creativity. The familiarity may help to clarify some outstanding issues but it may also cause fixation on the familiar interpretation. This occurs when the existing approach is re-used literally. Thus, the reference to a familiar design case may inhibit creative insight, as well as enhance it. Hence, the presence of past design cases is, in general, a helpful but not a sufficient condition for creativity.

Rather than the actual design cases, the process of their re-use is more important [WP97].

There are many models of a case-based approach to design, but there are fewer generic techniques for designing creative and innovative products. For instance, Altshuller developed a theory of inventive problem solving by generalising over a large number of cases accepted as innovative patents or inventions worldwide. His theory (known by its Russian acronym 'TRIZ' or English 'TIPS') is an algorithm for improving the chance of designing an inventive solution to a problem [Alt84]. TRIZ focuses the (inventive) design task into a small core of basic procedures. First, an explicit contradiction is identified in the specification of product behaviour. Next, the core of an invention is the removal of such a contradiction, where contradictions are mutually connected features of the designed artefact, and an improvement of one of them worsens the other one. The main contribution of TRIZ is its generalisation of many possible contradictions to a matrix of typical contradictory physical properties. Thus, the design is about associating a contradiction identified in the current problem with one or more contradicting patterns in such a matrix.

Altshuller also proposed a table of methods for the removal of the identified contradictions. These are called *inventive principles*, and they are also generalised from the past inventive solutions. Thus, TRIZ assigns one or more fixes (i.e., inventive principles) to the typical contradictions between the physical quantities and behaviours. Since TRIZ was developed for designing technological products, it refers to physically observable and measurable contradictions, e.g., if the design brief requires a particular component to exhibit properties of both conductors and insulators or be both present and absent.

The operation of recognizing a contradiction by means of retrieving applicable inventive principles is supported by a strong empirical evidence gathered from thousands of inventive cases. The evidence is condensed into a succinct form of a matrix. In the CBD terminology, the inventive matrix acts as a referential index for the recognition of certain patterns in the problems. The adaptation of an inventive principle to the current problem is also supported by examples; i.e., how other implementations of the same principle were designed, what structures they contained, etc. The help with the transformation of rather abstract inventive principles is less detailed. Consider the following sample problem from [Alt84]:

Problem: A lens is to be polished to a high precision and accuracy, and a coolant needs to be delivered between its surface and the polisher made of resin. An attempt was made to incorporate slits into the polishing assembly where water could be squirted, but the perforated surface of a polisher performed poorly...

Contradiction (verbally): The requirement for cooling conflicts with the ability to polish glass: A solid polisher cannot cool sufficiently, a perforated one does not deliver quality. Thus, the polisher must be both 'perforated' *and* 'solid' simultaneously...

Contradiction pattern: A structure exhibiting desired behaviour with respect to temperature vs. the accuracy of manufacturing

Inventive principle: Create a structure with the desired property (temperature) from an existing (form of) structure

Examples of the principle: [...] apply phase transitions, mechanical or acoustic oscillations, or consider cavitation.

Inventive solution 1: Make the polisher from a micro-porous material; this contains microscopic holes for pouring a coolant but is 'solid' at a macroscopic level. A porous polisher both is and is not solid → implements *Cavitation*

Inventive solution 2: Make the polisher from abrasive particles frozen into ice. As the lens is polished, the ice melts, thus delivering water (a coolant) on the surface. The polisher is solid but simultaneously, water can pass 'through' it → implements *Phase Transition*

The process of recognising and exaggerating a contradiction is structured and supported empirically, but it is up to the designer to find an implementation of a particular inventive principle. Although the designer must recognise what can be re-used from the inventive principle, s/he already works with a pattern (model) that needs to be instantiated. An exaggeration of a problem specification into a contradictory pair of parameters helps avoid an exhaustive exploration of all analogies between the past and current problems. The empirical matrix offers a conceptual solution, a model that removes the contradiction and may inspire novelty.

Several techniques for recognising conflicts were developed [Alt84, SMW95], but the process of recognising contradictions is not elaborated in depth. It may involve iterative steps refining the designer's perception of the problem and addressing the issue from different perspectives. Another gap worthwhile addressing is that unlike in the example above, the contradictions are usually hard to articulate in such clear and explicit terms. Thus, a designer or inventor may spend a significant time trying to elicit the essence of a tacitly perceived conflict before finding the one pushing the design forward. As

shown later in the book, TRIZ concepts can be incorporated in the recursive theory of framing in design. It may be perceived as one of several mechanisms that implement reflection on design actions.

3.2. Approaches to handling ill-structured problems

A lot of research addressed the poor structure, vagueness and ambiguity of real-world problems; design is no exception. This accelerated with the arrival of computers as problem solvers, where the vagueness and the need to work with incomplete information posed a major obstacle to algorithmic machines. Common sense became another major hurdle [McC80].

One method for handling incompleteness is the theory of circumscription [McC80]. This defines a schematic rule $\Phi(x)$ that conjectures that only the objects that can be shown to have certain property by explicit reference to given facts, are *all* the objects that have that property. For example, if, in the well known puzzle, there is a boat available for transporting the missionaries and cannibals across river and nothing else, the boat can be circumscribed as a sufficient and the only means for crossing the river [McC80].

When a common sense rules that a boat can be used for crossing the river "unless there is something wrong with it or something prevents its use" [McC80], circumscription can restrict the world to comply with it. In other words, if there are no explicit facts requiring that something prevents using a boat, the circumscription conjectures that there is *nothing* to prevent using it. Thus, no new objects (such as bridges or airplanes, in this example) can be introduced into the circumscribed world unless they were there before circumscription.

The assertion that a finite set of explicit statements sufficiently specifies a (design) problem is not a 20th century invention. Its origins are in the rule by William of Ockham articulated six centuries ago. Ockham's rule (or Ockham's razor) asserts that it is unnecessary to artificially conjecture more (objects, conditions) when fewer can explain the situation as well[1]. In other words, although we work in an open world, we attend to it using a minimal set of statements that are needed in a particular situation, and no more. This enables

[1]William of Ockham (1280-1349) – a philosopher and logician pioneering the modern a conceptual stance towards knowledge. His 'razor' is originally formulated as follows: "*Frustra fit per plura quod potest fieri per pauciora.*"

us to cope with the ambient ambiguity, incompleteness, and 'infinity' – circumscription is thus a modern version of an old wisdom.

Another attempt to reduce the conceptual world into necessary 'bare bones' led to the "*closed world assumption*" – saying that only the objects explicitly mentioned are objects that are needed; that is, only those objects exist in the conceptual world. An alternative articulation of the close-world assumption (CWA) is [CF82]: "CWA is that all relationships not explicitly stated to hold do not hold."

CWA and circumscription can be expressed in terms of minimal models and their completion. Accordingly, "if a sentence is inferable from a minimal extension of certain axioms, then it holds in all minimal models" [Dav80]. The essence of this theorem is in the minimal extension of an axiom that is defined in terms of relativising all formulas of the given theory. The formulas allowed in the relativisation are expressed by a predicate schema $\Phi(x)$ where x is a free variable. Thus, Ockham's rule, circumscription, and minimal model completions do nothing other than reducing a complex conceptual world into manageable, unambiguous, and *explicit* problem solving space.

Sometimes, this attempt to circumscribe removes the boundary between the worlds and their models. An example is the definition of ontology [Gru93]: "ontology is an explicit specification of the conceptual representation of a particular domain. The term is borrowed from philosophy where it is a systematic account of Existence. For knowledge-based systems, what 'exists' is exactly what can be represented." From a knowledge-level perspective, this is promoting the explicit representation of knowledge to knowledge itself. However, as mentioned earlier, there may be tacit knowledge that cannot be explicitly represented! So, the definition of ontology is only acceptable as model to deal with the incompleteness of the real world by closing its conceptual boundaries.

Circumscription or a conceptual closure of the world is an important operation for tackling the problems in a real world. On the other hand, design cannot be confined within a single conceptual world that is closed at the beginning. One may indeed reason about the problems only inside a circumscribed world, but at the same time, there has to be a chance to re-circumscribe the world to contain different concepts and objects. In other words, *design unites the processes of circumscribing and re-circumscribing* the world, and it is therefore interesting to ask how a re-circumscription can be accomplished.

3.3. Knowledge and reflection in action

I touched on epistemological difference between a possessed knowledge and knowledge applied in action in section 2.4.4. Before showing how this links to the incompleteness issues, consider the following extracts describing the design activity [Sch83]:

> Because of the complexity (of design), (...) the designer may take account of the unintended changes he has made in the situation by forming new understandings and making new moves.
> He shapes the situation, in accordance with his initial appreciation of it, the situation 'talks back', and he responds to the situation's back-talk. Design is thus a conversation with the situation.

Schön's quote highlights the presence of 'shaping' in design; during a 'shaping operation' the designer forms a point of view for further investigation of the problem (design situation). He or she imposes a particular view on the problem to solve it and to learn more about it. Compare this imprinting of an initial appreciation to the circumscription: Schön's 'shaping' is an enclosure of the world as discussed in section 3.2. However, Schön claims that a situation can be re-shaped (i.e., re-circumscribed), e.g., triggered by unexpected results arising from the initial 'shaping'.

Shaping and re-shaping of the design situation must be *a reflective process* [Sch83]; the designer has to *reflect in-action* on his or her current understanding of the problem. This includes tacit appreciation of the strategies used to support a particular decision, which, in turn, helps to address the phenomena or features implicitly present in the performed action. Such features only became apparent after a retrospective reflection. In other words, through reflection it is possible to see the effects of the designer's tacit or implicit understanding on the shaping of the problem. Through reflection it is possible to criticise tacit understandings formed by a repetitive experience and practice of designing. The quoted *conversation with a situation* is, therefore, an interplay of shaping the situation, reflecting on it, and re-shaping it. This cycle can be repeated until its results conform to the designer's tacit, empirically derived expectations.

Design 'shaping' (or *framing*) may be seen as a specific kind of circumscription that restricts two mutually linked conceptual worlds. The designer's frame circumscribes his commitment to certain objects, structures and models used for constructing a solution. Vice-

versa, particular solution objects circumscribe the changes of the commitment to a certain interpretation of the problem. The conversation of problem solutions and problem understandings meas that the two commitments cannot be made independently. If a designer attends only to the development of solutions, the design process is crudely simplified; cf. the example of shawl weavers using hand-made and artificial dyes in section 2.3.1.

A similar notion of conversation can be found also in [Fis92, FNO93, NSH94, NYS+98] – this collection is discussed in section 4.1.5. The advances in requirements engineering (RE) reported in [Nus01] are also close to the perspective of shaping and re-shaping presented in this section. The evolving nature of design specifications is an essential part of modern software development [dGRNK99, NE00, RNK99]. The 'Twin Peak' model for requirement engineering captures the mutual interactions between architectures (solutions) and requirements. Thus, the principle shown in Figure 3.3 illustrates the *reflection-in-action* when deciding on valid design requirements.

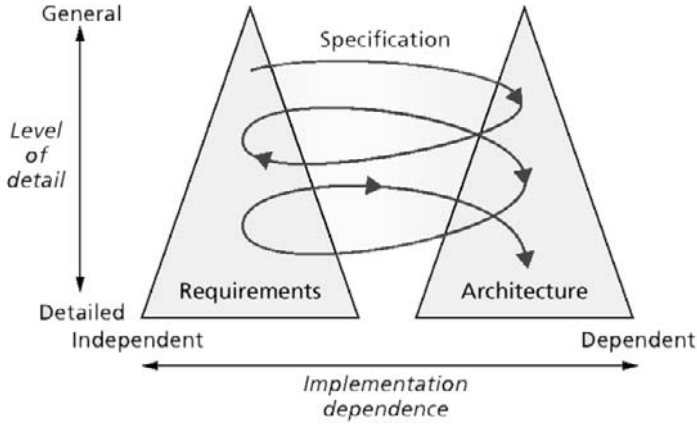


Figure 3.3. 'Twin Peak' model illustrating reflective conversation with design situation (based on [Nus01]).

Another approach addressing the interactions between different conceptual categories is the "Function-Behaviour-Structure" (F-B-S) framework [Ger98a, Ger98b]. As Figure 3.4 shows, design begins with a desired functionality F; accordingly an expected behaviour of the product B_e is articulated and appropriate structure S is chosen.

Sequence $F \rightarrow B_e \rightarrow S$ is thus *a solution synthesis*. An important facet of the F-B-S model is that the re-formulation runs backwards from structure S through the actual behaviour B_s, it aims to amend the required functionality ($F \rightarrow F'$) and/or the expected behaviours ($B_e \rightarrow B'_e$)[2].

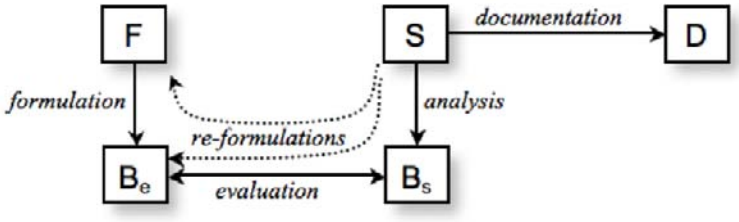


Figure 3.4. Function-Behaviour-Structure (F-B-S) model of design [Ger98b]

According to the F-B-S model, reflection is closely bound with the situation; it matters when, where and how a particular reasoning chain occurs. The *situatedness* is manifested in design by the selection of an appropriate behaviour and structure – this is not random nor is it given a-priori; it responds to the design situation and depends on the context, perspective, and focus. The interpretation of a situation depends on how it is represented, and vice-versa, its representation is influenced by the interpretation.

3.4. Summary or what is the design about?

Perspectives reviewed in this chapter are presenting partial aspects of design, and as such have their specific weaknesses and strengths. In this section, the reviewed approaches are put together and possible associations between the different views are explored. The key outcome is a list of typical features that, in my opinion, characterise engineering design. This list is used in the subsequent chapters to develop and validate my recursive model of framing in design.

First, we touched on the difference between the search and exploratory paradigms in the active or passive role of the designer. In

[2]The last concept is labelled as D and stands for a design documentation, justification, and explanation of a particular design choice.

'design as search' the agent has all the necessary knowledge all possible design elements, structures, modules and relationships among them. This space can be vast and only a small part of it constitutes satisfactory and/or optimal design solutions. In this perspective, it would be sufficient to develop fast techniques for navigation in such a vast design space, and all design tasks would become routine.

Unfortunately, this is not the case in the real world. Search techniques work for relatively well-defined problems, but design problems resist the application of accelerated search techniques. As a workaround, some a-priori or heuristic knowledge may help the agent to distinguish between a dead end and potentially fruitful design paths. The introduction of a heuristic enabling design space exploration naturally extends the paradigm of search; however, the exploratory paradigm as a realistic enhancement of search does not actually live up to the promise it makes.

A typical source of heuristics is the designer's experience with the similar design situations. Nevertheless, many theories leave untold the process, by which the heuristic control is exercised. In most cases, heuristic reasoning is ascribed to the designer, who acts as a kind of 'black box'; we often refer to personal experience and empirical knowledge without going into any details. This observation suggests that in addition to the explicit design space there is also another knowledge type that cannot be positioned in the same design space and is used to accelerate explicit search strategies. This *tacit knowledge* cannot be articulated in terms of those well-defined, conceptual design spaces.

Similarly, one can construct simpler or more elaborate operational models of design. As long as the criteria for testing proposed design solutions are known in advance, and the proposed artefacts satisfy these criteria (albeit to different degrees), the various models perform satisfactorily. If an explicit criterion is not satisfied to the desired degree, known fixes enable designers to modify the current solutions. Some models [Ull97] acknowledge that new criteria may emerge during design, but they address the issue of emergence only superficially. A common source of new criteria is 'a third party' (for example, a customer) that evaluates the design and formulates additional criteria or requirements for the design problem specification.

A customer may reject a design solution, which is perfectly optimal according to all given requirements, constraints and norms. The

same may happen also without any explicit customer acting as a judge. The designer may reject a sound solution, because "*it just does not seem right*" [Sch83], however, such reasoning 'turnarounds' are beyond the scope of any operational model of design. Such turnarounds are common in design, and may be due to the incompleteness of design situations [Sch83]:

> Because of the complexity, the designer's moves tend to produce also consequences other than those intended. When this happens, the designer may take account of the unintended changes he has made by forming new understandings and making new moves. Design is thus *a conversation* with the situation...

This quote mentions one reason for a turnaround – observing unexpected or unintended behaviour. Designers typically express their dissatisfaction with the partial solutions in terms of the solution 'not seeming to be right' [Sch83]. This assessment against their tacit expectations is followed by a reflective attempt to articulate the gap in explicit terms. Although the tacit knowledge informing the decision about unacceptability remained unexpressed, it was applied in the solution evaluation and in re-shaping of the design situation that, in turn, facilitates the "*conversation between the designer and a design situation*" [Sch83].

Another issue arising from the operational models is a crisp border between the design phases. Although Ullman [Ull97] speaks against the over-the-wall design, many models still attempt to separate the indistinguishable. Strict boundaries among design operations contradict the idea of "*a dialogue with design situation*" [Sch83]. Practitioners understand and analyse the design problems not per-se, but through the attempts to change (i.e., synthesise) its solutions. Vice-versa, the solutions are refined following a sequence of attempts to understand the design situations [Sch83]. Thus, *a tangled co-evolution*, an interplay of problem specifications and solutions, rather than a crisp and a stepwise process models what is happening in design. Moreover, designers often work with incomplete and inconsistent knowledge. Instead of avoiding the inconsistency thus severely restricting their creative freedom, they iteratively probe and articulate their intentions by testing out partial solutions. Thus, a *conversation with the design situation* [NSH94, Sch83] or *intertwined development of specifications and solutions* [Nus01, SB82] are the essential features of non-trivial design problems.

Cognitive models of design consciously acknowledge the existence of non-explicit knowledge possessed by the design practitioners. This knowledge comes to the fore in discovering new aspects in the current design situation, e.g., by means of a designer's insight into the problematic situation [DD95], or by means of creative leaps and firm standing among familiar ideas [CE96, Cro97]. One cognitive model attempts to construct models of this insightful, non-explicit and non-monotonic forms of reasoning [Cro97]. This model proposes methods for how a new solution may appear as a result of manipulations with the familiar ones or with the solution models. In a similar way, problem specification may also need to be manipulated, and this may have a bigger impact on the designer's insight than any solution modifications. Designers reason about the modifications drawing on their creativity and experience from the past exposures to similar tasks. However, at the bottom of many creative and insightful designs is knowledge complementing the explicit reasoning processes. On the other hand, references to creativity, insight,and deep domain knowledge only add more complexity: Are these concepts *the reasons* for the designer's expertise in practising design, or are they *the consequences*? If they are consequences, then there seems to be something left unattended by the cognitive models I reviewed.

Next, we looked at creativity in more depth, and considered TRIZ [Alt84]. This theory adopts a procedural stance towards creativity and formulates an algorithm for the resolution of non-trivial problems. This attempts to merge explicit rules and principles of designing with experiential knowledge, which, in turn, helps to uncover the unacceptable cores of the design problems. This core consists of an exaggerated contradiction of the physical requirements on the designed systems. A specific contradiction characterising the problem is then matched to an empirically obtained reference index, to find applicable solution models ('inventive principles').

Techniques proposed in TRIZ to address an explicit contradiction by constructing an abstract solution model are particularly useful for non-trivial designs. Also, in designing innovative products, previous cases often act as an inspiration for the resolution of identified contradictions. The process of articulating the exaggerated contradictions may involve several iterations and explorations of past solutions. However, TRIZ leaves the whole reasoning in respect to how a particular inventive principle can be implemented to the designer

and his domain expertise. An important contribution of this model is in its intimate relation with the past design solutions. The aspect of shaping the understanding of the design problem by constructing applicable solution models is clearly embedded in this theory. Also, the process of shaping is recognised in the form of an articulation of explicit contradictions.

Capacity to see analogies allows understanding of the vague design situations and developing unique design solutions. I emphasise the interplay of both – *seeing-as whilst doing-as*. In other words, design requires an active action, an experiment or a solution construction to verify the perceived analogy. Instead of a simple "*How to do it?*", design is about a combination of "*How to do it?*" and *What to do?*" These two questions are usually followed by a speculative but important "*What if?*", i.e., an experimental probe into a design situation. Often, an interplay of these question is the key to the discovery of something new that was previously overlooked.

Thus, tacit knowledge and its use in probing by means of analogies is an interesting form of meta-control of the design process. It helps to cope with incomplete and underspecified design situations, and at the same time prevents the designers from generating too many conjectures. Design takes into account the designer's subjective expectations, personal attitudes, assumptions, intentions, and overall impressions from the proposed solutions. In other words, the assessment of the logical validity of designed products is needful, but it is not sufficient for designing! There are also 'non-logical' means playing an important role in design, and, in my opinion, these means deserve a place in the design models.

Let us conclude this chapter with a list of tenets that, in my opinion, characterise design:

- Design develops new artefacts that satisfy desired goals.
- Design is inherently an iterative process.
- The understanding of the 'desired goals' changes during design.
- Design relies on the exploration of consequences through conjectured experiments and partial solutions.
- Design uses a variety of knowledge sources including designers' past experience, and analogy and similarity are important vehicles for 'seeing as' and 'doing as'.
- Multiple parallel lines of enquiry are open during design.
- In addition to rigorous assessments, a subjective feel for the

situation is also a driving force in non-trivial designs.

These characteristics of design are applied in the construction of my recursive model of framing in chapter 5, as well as in the evaluation of my supportive evidence (chapters 6 and 7). Next, I review a few of the existing implementations of the knowledge-based design support systems. Then I will conclude the following chapter with an articulation of the gaps between my perception of design and that of the existing approaches.

Chapter 4

Knowledge Intensive Design

So far, I considered engineering design at an abstract level from different perspectives, and highlighted the relationships between these perspectives. It was argued in the previous chapter that reflective knowledge used in the reasoning processes in design is a natural extension of the explicit knowledge of design elements, methods and practices. This tacit knowledge was typically manifested and criticised in the reflective re-shaping of design situations. The objective of this chapter is to review selected efforts implementing knowledge-level models of design. In the review I focus on the design support applications and how they address the gap between the initial, vague expectations, and the formal specification of a design task.

As it was the case with the definition of engineering design task, there are several views on what is meant by 'knowledge-intensive' or 'knowledge-based' design tools. Different definitions divide into attempts trying to 'get the human out of the loop' versus attempts trying to 'get a computer into the loop' [Fis92, Gar98]. From this point of view, it is possible to review roughly three categories of design support tools:

1. *Automated design systems* (also known as intelligent CAD or I-CAD), whose aim is an automation of design: once expert knowledge is acquired and encoded in the I-CAD system, the human designer only acts as a source of additional requirements or new evaluation criteria [Mac90];
2. *Knowledgeable design assistants* are able to assist the human designer in one or more phases of a design process and typically have an extensive problem- or domain-specific knowledge to re-

call and to adapt similar design cases [WP97]. Often, they have to maintain multiple design contexts and do routine reasoning [SCD+90, Tan97];

3. *Knowledgeable design critics* usually follow the formal development of a product or the informal justification of a designer's decisions; they tend to draw on a rich domain-specific knowledge to critique or to comment on the explicit patterns of design [Fis92, NSH94]

I will focus on the last two types, as in order to take the full advantage of tacit knowing one needs to be involved in the design action [CB99, Sch83]. An explicit formulation of a designer's understanding of a problem is, generally, insufficient to guarantee a proficiency in a particular action. The idea of placing a human into a role of 'Big Brother' commanding a support system by an explicit formulation of requirements and criteria violates this active role. It inhibits the human ability to increase their expertise and to acquire deeper understanding of the design problem. I support the view that automatic programming is *not achievable* when the goals are articulated outside the problem solver – it is also *not desirable* because humans enjoy doing, in addition to observing others doing the job [Fis92].

I see knowledge-based design support systems as enabling tools have the ability to enhance the power of human designers and support some reasoning processes underlying design. Hence, I use the following definition for a knowledge-intensive design support [Tan97]:

> [Knowledge-intensive design tool] is a computational system that supports decision making during the design process, and enables designers to explore the structure of design problems and their solutions. It combines human designer's (tacit) experience with (explicit) knowledge of the domain that may be stored in the tool.

If we see design support systems as enabling technologies for the exploration of the problem structure, we can demand additional properties for such systems. In particular, to make a design support system useful, one important property is its capability to adapt, to acquire new knowledge. Adaptability is one of the most significant human traits [Sim69]. People adapt to the given problems by articulating new conceptual frames and by interpreting those problems. Any support tools that aim at enhancing human powers should also exhibit similar adaptive capabilities. In this sense, one may even call such design support systems 'intelligent'...

What I review next are typical representatives of the above categories. In particular, I look at the roles knowledge models play in design support. My discussion of knowledge-level models in design starts with the significant outcomes of selected projects contributing to this field.

4.1. Overview of design support philosophies

Since I give a short account only of selected approaches to design support, it does not mean that the absence of any approach from this overview makes it less important. Many different approaches were looked at, and instead of exhaustively reviewing all of them, I opted for a few representatives of their classes:

- *Edinburgh Designer* is an example of a blackboard and truth maintenance system for exploratory design;
- *KRITIK* family illustrates the case-based design support and the opportunities provided by the use of domain models;
- *IMPROVISER* is an implementation of a design model using case-based and constraint satisfaction approaches;
- *Invention Machine Lab* is the implementation of TRIZ (see also section 3.1.4 for the principles of the theory);
- *DODE-s* feature several domain-oriented design environments implementing the design rationale, critiquing and commenting.

Each of the tools is reviewed with respect to the tenets that were articulated in a conclusion to the review of different perspectives on design in section 3.4. Hence, one may consider the review of the tools as validating the identified properties of the design process.

4.1.1. Edinburgh Designer

The Edinburgh Designer System (EDS) is a knowledge-based design support system developed at the University of Edinburgh [LCS92, SCD+90], or better, it is a series of developments in the field of knowledge-based support for design over several years. EDS embodies the knowledge-level view of engineering design that treats design process as an exploratory task. In EDS, a design task starts with initial requirements; these are later refined into design description documents (DDD). The construction of DDD-s is iterative and interactive in terms of knowledge that is generated in the process. Such

generated and acquired knowledge is stored in domain knowledge base (DKB), which bootstraps the design process by identifying the regions of interest in in a particular domain.

DKB contains knowledge in the form of clichès that can be see as generic knowledge-level models of elements, modules and relationships in a particular domain. The clichès are closer to knowledge frames [BF82] than to design prototypes [Ger90], therefore they may contain also knowledge of 'doing design'. Clichès may explain how to use a particular module or calculate a particular parameter, and are organised in hierarchical structures, thus linking to the work on common ontologies [Gru93] and knowledge-level models [New82].

Conceptually EDS is about the exploration of the space of potential designs. This is achieved by regarding the designers' statements and decisions about the design as *assumptions*. Term 'assumption' is taken from assumption-based truth maintenance systems (ATMS) [dK86a], and denotes currently believed statements about the problem, the world, etc. Assumptions are distinguished from the axioms in the sense that the belief in axioms is unchanging, whereas the belief in assumptions is subject to change and re-validation. ATMS maintains parallel lines of enquiry that represent alternative solutions, and may be mutually incompatible. The alternative directions are treated as assumptions and ATMS divides clashing assumptions into independent *contexts*.

Clichès are an important concept from the knowledge representation and presentation perspective. They contain definitions of various classes of devices and components in a particular domain, various applicable parameters, constraints and descriptive constants. In addition to the declarative and parameterised relations, they contain knowledge of how to assemble them. Consider a simplified example of knowledge representation in EDS (adapted from [SCD+90]):

Class name:	water turbine
Parameters:	Initial/given:
	Calculated:
Variables:	generator efficiency,
Constraints:	optimum speed of model is 70 rpm
How-to's:	see functionTable

EDS has a blackboard architecture that controls the calls to different knowledge sources depending on the sets of underlying assumptions and relevant operations. A new datum can be inferred using

any of the available knowledge sources, including module definitions, geometric, algebraic or clichè engines. The designer is also modelled as *one* of the knowledge sources, and as such has to compete with the other sources to contribute to the reasoning process. The designers are in the context of EDS a typical source of design assumptions. If a designer's contribution is activated as a knowledge source, EDS accepts the decisions and propagates the consequences of such external decisions in DKB and in active DDD-s. The same propagation takes place with a contribution coming from any other source (e.g., algebraic calculation).

Generally speaking, EDS is a representative of a large class of engineering design support systems that feature multiple knowledge sources, opportunistic control strategy, co-operation between computational tool and human designer, consistency maintenance and parallel design contexts [Tan97]. These are all aspects complying with the characteristics of a design process from section 3.4. However, some shortcoming not tackled by the EDS approach include, among others, the unclear role of the designer's meta-knowledge and tacit knowledge in designing artefacts. Although the designer is treated as one of the knowledge source, unlike other knowledge sources in EDS, the designer's comments are never rejected or critiqued. If they violate the existing contexts, a new context is created and the DKB is respectively re-organised.

Such flexible behaviour is good, but it is also the source of many sudden decisions that are not explicitly grounded in the active design context. To a certain extent, they may reflect a designer's tacit knowledge; however, the designer remains unquestioned. EDS is not supporting any form of further exploration and probing in the DKB, or in the alternative directions that are implicit in the DKB. It does not support conjecturing unusual approaches to the problem, apart from propagating the assumptions. Thus, EDS refines the view shaped by the designer's assumption and performs routine reasoning in the explicated DKB. However, it lacks a method or perhaps a specialised knowledge source able to drive or justify such a shaping.

To summarise, blackboard-based systems with an opportunistic control flow are a significant development in the field of artificial intelligence used in design support. In chapter 5 I further elaborate and extend the notions mentioned here, include the opportunistic control and dynamic design contexts (frames).

4.1.2. KRITIK, IDeAL and family

This section is concerned with systems developed at Georgia Tech over the course of several years. The review highlights the essential concepts and details can be found, e.g., in [BG94, BGP94, PG98]. Although KRITIK is the 'founder' of the family, the review is based on the advanced design support tools IDeAL and E-KRITIK that inherited many features from their ancestor.

The above tools are integrated case-based and model-based design systems utilizing analogy-based retrieval of similar cases solved in the past from a memory of cases. The retrieval engine uses richly indexed base of past designs, and as the key index it uses the functionality delivered by a particular solution. In addition to the actual functionality, each case contains a description of the underlying structure and its behaviour(s). Causal behaviour links the functionality with the structural elements forming the solution. Once a case functionally analogous to the current problem is retrieved from the memory, the system focuses on its behavioural model. The behaviour of the analogous case is revised to incorporate any particularities of the new problem and becomes a skeleton of the new design solution.

New design solutions are verified in a qualitative simulation; i.e., the system tries to infer the desired functionality from the particular structure and behavioural links. The IDeAL tool extends the validation and is able of simple learning: New designs (i.e., their structures and behaviours) are cross-indexed with the existing ones. If a large number of similar problems is stored, IDeAL can learn emerging commonalties in the stored records. The stored records are represented as triplets called Structure-Behaviour-Function (S-B-F) paths, or alternatively, F-B-S paths [QG96], as design typically starts from the desired functionality, and structures are designed (e.g., abduced) accordingly. An example of this representation, adapted from [BG94], is in Figure 4.1.

The family of these systems draws on early ontological representational schemas of components and substances [BC85], and extends them into S-B-F model. For example, terms 'WATER' or 'HEAT' in Figure 4.1 are substances and 'H2SO4-pipe' is a component containing the substance. Components are usually composite structures containing various substances and having various relations to other components. Functions are special schemas that relate the state of a

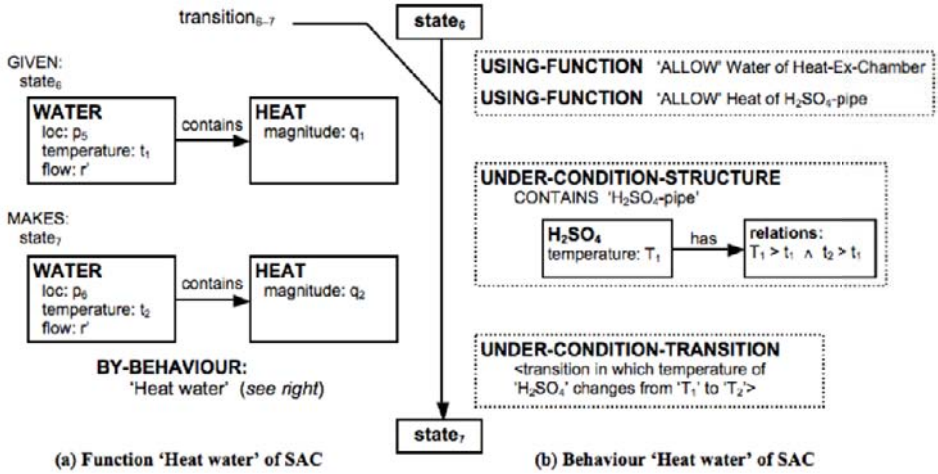


Figure 4.1. An example of an S-B-F representation (based on example in [BG94]).

component at its input with its output state through a causal chain of behavioural transitions. For instance, the function of the device in the figure is described as a change in the water temperature ($t_1 \rightarrow t_2$) by the absorbed 'HEAT' flowing between two locations p_5 and p_6.

Since the functions are described as relations between input and output, they may be organised according to their type (e.g., transformation, movement, etc.), and such a typology then serves as an index. Thus, statement 'USING-FUNCTION ALLOW...' on the right of Figure 4.1 refers to a class of devices that allow the presence of a particular substance (e.g., water) in a particular component (e.g., Ex-Chamber). Behaviours are state transitions between the states of a device or component; typically, they are causal relations describing how the transitions follow each other. An example of a behaviour for heating water by absorbing heat is schematically represented on the right of Figure 4.1.

Model-based approach is also realized in a specific context: *Model-based hypothesis formation* is essentially an application of the knowledge of functional typology and of known structures that may be generalised. The result of generalisations is the creation of a hypothetical design case, a model scenario that subsumes many concrete design cases. Such hypothesised models can be later re-used for developing new devices functionally similar to the learned models. This

approach to prototype formation is also supported by [Ger98a].

One improvement of the notion of S-B-F paths includes interactions with the environment of a designed artefact, because a common motivation triggering design is that an existing device is modified to fit a new environment. This, in turn, introduces the notion of artefact adaptability to the changing external conditions. Environmental interactions are modelled in parallel with the internal behavioural transitions, and they are often justified by generalised model patterns. E-KRITIK is thus an environmentally aware spin-off from KRITIK relying on the model-based engine, which it uses not only for the generalisation of design cases but also for reasoning about behaviours and for distinguishing the intrinsic functions of a device from the environmentally-bound ones. The former functions are abstractions of the complex internal behaviour of device components; the latter function is an abstraction of the interaction of a device with its outer environment. This 'environmental' view on device functions also upholds the social nature of design processes [QG96].

4.1.3. IMPROVISER

Another case-based tool for design support is IMPROVISER [KW96], a model and its implementation based on the cognitive aspects of design. Methodologically, this system subscribes to similar models of design as those in section 3.1.3; especially, the recognition of an artefact's emerging feature or its suitability for a certain purpose. "*What is noticed as being relevant not only influences what new variables emerge but it may also determine how an object is used*" [KW96]. The sudden and unexpected recognition of a new object often involves the discovery that ordinary and common objects have new functions or serve new purposes. Such a gradual emergence of design criteria, parameters, requirements and constraints is then an essential feature of supporting innovative design.

According to the IMPROVISER model, the focus is on the preparation phase of design. This is the divergent phase, when the problem is viewed from different angles (see also section 3.1.3). New ideas may emerge in this phase, or existing constraints may be relaxed and modified and the problem may be re-formulated and re-interpreted. What is evolving in this phase is mainly the problem specification; however, it happens in parallel with recalling potential solutions and

past cases. Preparation is followed by the assimilation of discovered features, when conflicts may arise within the current perspective and the adaptation of the proposed solution must be performed. Assimilation phase feeds back to the preparation and may return the design process back to investigate the emerging issues.

The implementation contains several modules dealing with and supporting different cognitive actions. At the bottom is the long-term memory containing a library of previous design cases. Tightly bound to it is the working memory managing the design options under consideration; relevant designs are stored as problem contexts and serve as one of the inputs for the solution transformation module. This module is able to perform various cognitive operations upon design contexts; e.g., merge them or augment them with the experimental data. Design specification evolves by manipulating the existing knowledge in the situation assessment module. This has the means to discover opportunities for re-interpretation of the current findings and for generalisation of empirical data. Overall control is left to the blackboard architecture (similarly as in section 4.1.1).

Core of the IMPROVISER is *a problem context* representing a sub-problem of an overall design problem and containing specific knowledge and constraints. Contexts are organised along two complementary dimensions: First, sub-problems are refinements of higher-level problems; a complex problem is decomposed into simpler sub-problems that may be tackled separately. In addition to vertical links, sub-problems may exhibit horizontal, i.e., peer-to-peer relationships. Sub-problems may share parameters and influence them in a contradictory or co-operative manner. For instance, as illustrated in [KW96], the egg launching sub-problem of a larger plant tends to increase the launch strength of a spring, which increases the plant capacity. However, the faster the eggs move, the more likely they crack; therefore a cushioning sub-problem tends to restrict the velocity of the eggs. The two sub-problems contradict each other with respect to the egg velocity and plant capacity parameters.

Each context is described by a set of considered design options, cases and partial solutions. It also contains a set of conditions the solutions must satisfy, an index between the options and conditions regarding the degree of their satisfaction, and finally constraint priority schemas. Contexts are created dynamically as the proposed options are considered and the problem specification evolves. If the system

unexpectedly finds a relevant feature or an existing case addressing the issue in focus, it may add it to the current problem context, and the context may need to 'assimilate' new information. Thus, the role of problem contexts in IMPROVISER is similar to the operations of problem shaping and re-shaping (section 3.3).

From the given sub-problems IMPROVISER may generate a dependency graph of their parameters, draw the designer's attention to potential conflicts, and apply the fixes associated with the conflicting constraints to find a better assignment to a particular parameter. From this perspective, this tool behaves as a typical constraint checking and propagating engine [Kum92]. Thus, it represents another variant of hybrid systems, similarly as, e.g., EDS from section 4.1.1 or KRITIK from section 4.1.2. Unlike other tools, IMPROVISER uses past design problems to formulate additional constraints. Constraints from the sub-problems are propagated throughout the dependency network, and fixes are identified in respective problem contexts to resolve conflicting linkages. IMPROVISER is one of the most advanced implementations of cognitive modelling in design. Unlike others, IMPROVISER addresses also the 'woolly' terms, such as sudden insight or problem shaping (framing), and acknowledges the existence of contradictions in problem contexts.

4.1.4. Invention Machine Lab

One successful application of the theory for inventive problem solving (TRIZ) [Alt84] is the set of tools marketed by Invention Machine^{TM} Corporation as IM Lab or as TechOptimizer. Other implementations and uses of TRIZ are described, e.g., in [Shu99, SMW95][1]. IM Lab pays particular attention to the systematic articulation, representation, and utilisation of design knowledge. Particularly important is the knowledge of manipulating the elements, cases and models known in a particular domain. The essential knowledge source for the tool is a repository of inventions and patents (as described in section 3.1.4. From this repository, a few conceptual assumptions can be made [Alt84]:

1. Technical designs show domain-independent regularities;
2. Technical systems evolve by eliminating the identified conflicts;

[1] More information is on the community web site, http:// www.triz-journal.com

3. Any inventive problem can be represented as a contradiction of some parameters in a respective abstract technical system;
4. Knowledge of physical principles in the abstract technical systems can be used to identify and resolve the contradictions.

To address the first two tenets, IM Lab's Inventive Matrix aggregates typical combinations of various effects represented as abstracted patterns of contradicting parameters. A contradiction is the key to the system, which, in turn, provides several means for its removal and resolution. IM Lab support such a conflict identification by first allowing the designer to define the problem in the TRIZ-friendly, structured format. From the list of known parameters, such as speed or energy consumption, the designer selects a pair containing the desired feature, which is to be improved, and the contradictory feature (to be avoided). Having narrowed the design problem to fundamentally conflicting requirements, the IM Principles modules aims to resolve these conflicts and find an alternative problem definition rather than compromising design by retracting one the conflicting requirements. A number of principles is proposed to resolve the conflict, supported by examples of how they have been exploited previously. Each example has a rough sketch and respective behavioural description available.

Consider the example in the IM Lab demo: A hot bulky part is craned into the oil bath to harden, but being hot, the oil catches fire. This must be avoided. A simple solution is a lid with a hole in the middle. However, this makes it difficult to manipulate the bulky part; ideally, the lid should 'disappear; because it hinders the craning. This specific contradiction can be expressed in standard terms: Oil burns because its surface is exposed to high temperatures; when the surface is covered by a lid, this makes the manipulation difficult.

IM Lab offers three *abstract solution models*: (i) segmentation, (ii) vibration, and (iii) phase transition. Assume we opt for the segmentation and inspect past solutions applying this principle. Adapting the abstract 'segmentation' model one may arrive at a solution where the rigid lid is replaced by a layer of non-combustible granules (Figure 4.2). These cover the surface and, at the same time, give way when the bulky part is lowered into oil. The granules may also implement the principle of phase transition, if their material in a contact with fire released some extinguishing gas.

The IM Effects module associates various functions of the technical systems with the respective physical, chemical and spatial effects.

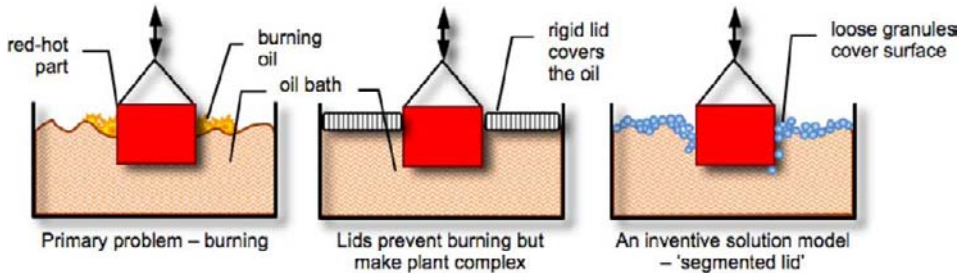


Figure 4.2. Illustration of sketches from IM Principles of the IM Lab in action.

Similarly as with the principles, effects are illustrated by examples of how they were exploited in the past designs. For example, the above-mentioned function to cover the surface of a liquid by segmentation is linked to the effect of observing the designed system at macro and micro levels. At macro level, the liquid is covered, at micro level there is only a dense array of loose particles that easily give way under minimal pressure.

The last module, IM Predictions, compares the current state of the solution with the ideal state. System ideality is the main law of the TRIZ: any technical system evolves so that the same function is delivered by simpler and cheaper designs[2]. This module is able to predict how a particular constellation of different components influences the performance of the whole system.

The main contribution of IM Lab-like tools is their focus on the correct problem definition. IM Lab does not try to solve design problems automatically, but by forcing the designers to adopt a structured approach, it does develop re-usable techniques and a vocabulary for the problem definition. The knowledge base of the IM Lab is valuable itself, due to an extensive range of principles, effects and examples of their past implementations. IM Lab can be helpful with its structured and systematic approach to design problem specification and exploration. However, it is up to the designer to reveal conflicts in the design problem, and to interpret them in abstract terms. Similarly, abstract solution models are exemplified, but the designer has to have sufficient knowledge of the domain to translate the principle or effect into the vocabulary of the tackled problem. This was

[2]Simpler and cheaper in TRIZ relates to the resources needed to deliver and maintain the function, including energy, material, or information.

illustrated above, when the concept of 'a segmented lid' was understood abstractly and brought in new structures (granules) that only behaved as a lid.

4.1.5. DODE-s or domain-oriented design environments

Researchers from University of Colorado at Boulder developed a series of design tools known as domain-oriented design environments or DODE [Fis92] for a range of domains. For instance, VDDE supports user interface design [NSH94, SBHK97], JANUS and KiD support interior layout design [FNO93], and IAM-eMMa and ART support image and document authoring [NYS+98]. Rather than reviewing the tailored environments separately, I highlight the similarities that are invariant across different domains.

Generally speaking, DODE-s are so-called *knowledgeable design critics*. The main purpose of these tools is to complement the designer's efforts, and assist with the uncovering of implicit features in the design situations. Design critics augment the designers' capability to evaluate their designs and to make decisions about what to try next [NSH94]. DODE-s stand in opposition to knowledge-based design assistants, the representatives of which were reviewed e.g., in sections 4.1.1 and 4.1.2. The main criticism of the automated design assistants is the lack of their reusability across problem domains due to their task-oriented, generic knowledge bases [Fis92].

The lack of deep domain-specific knowledge limits the amount of support given to designers, and hinders the communication between designers and clients. DODE-s aim at this gap by the integration of describing the problems and problem solving. In DODE philosophy, professional practice of designing has as much to do with defining and understanding a design problem as it does with solving it [Sch83, Fis92]. To this effect DODE-s employ various *languages of doing* for the representation of initial intentions (e.g., sketches, mock-ups, etc.). Such languages also simplify the reflection on the proposed solution, as the languages of doing feature a representational talkback, i.e., representations become able to reveal otherwise implicit design features [NYS+98].

Different environments, such as KID, JANUS, VDDE or ART, exhibit differences in the implementation and realisation of the individual modules. Nonetheless, a generic core of the DODE-s contains the following modules:

1. a high-level input interface with an appropriate language of doing for formulating initial requirements (e.g., KiD has a visual editor of floor plans);
2. a catalogue of components permitted to be used in design for the construction of solutions, often using iconic representation of components (e.g. in KiD the components are various types of stoves, sinks, fridges, etc.);
3. a simulator and analyser of design solutions for checking their compliance with requirements, guidelines and past cases, and for critiquing designers' actions (e.g., constraints for left- and right-handed kitchen users are checked automatically);
4. a case-based retrieval of past designs (in KiD, the past cases are kitchen layouts that can be translated into a visual language of doing and displayed);
5. a tool for recording and browsing argumentative justifications of design decisions (e.g., when overriding a constraint of layout optimality the designer must explicitly explain such a decision)

DODE case studies show that users often disagree with a criticism coming from the design tool, e.g., because of an unusual position of a sink in the kitchen. However, this disagreement is not harmful; different (and often strong) opinions usually trigger a deeper analysis of the reasons of the unusual decision. In many cases the disagreement leads to formulating counter-arguments and preferences, which evolve to new constraints and requirements on the compatibility of deployed components. The introduction of new requirements or constraints in response to an informed critique, or the modification of the existing ones, can be perceived as a shift in the designer's perspective. Procedurally, it corresponds to the operation of design re-shaping [Sch83].

Another design feature of DODE-s reveals the tentative nature of how the human designers work with vaguely defined artefacts. The visual construction tool and the associated log of justifications in DODE-s fit well with the vagueness of the design process, because the designers can test their ideas as soon as they express them in a suitable language of doing; e.g., as a sketch [Goe95]. Furthermore, each modification to the visualised solution sketch is assessed by the tool's analyser for its compliance with the domain constraints. Thus, the interplay of hands-on problem specifications and solution generation, as mentioned in section 3.3, emerges from such a co-operation between the tool and the designer [SB82].

To summarise, DODE-s represent a milestone in human-centred design support. They successfully address many issues associated with the cognitive complexity of design and designing, including the ongoing problem formulation and understanding, the co-evolution of problem specification and implementation, and the frequent usage of informal design rationale. If one compares these features with the requirements set out in section 3.4, there is a significant overlap, which, in turn, validates my conclusions in chapter 3.

4.2. Model-based design support

Let me next investigate the roles played by the knowledge-level models in design. In the following sections I return to knowledge-level models, two specific representative of this philosophy in design, and review their properties that are relevant to the tenets of non-trivial design problems defined in section 3.4. My discussion so far made several references to various forms of modelling of the designer's knowledge. Different roles of models were implicitly mentioned, and this section looks at knowledge models and their role in non-trivial design more in depth. Before reviewing two typical representatives of knowledge modelling in design in section 4.2.2, I touch on the notion of knowledge models and ontologies in general, in section 4.2.1.

4.2.1. Introduction to modelling on the knowledge level

Term 'knowledge level' was coined by Newell to describe a higher level of information processing in addition to the lower, symbol level [New82]. Both levels refer to different processes a knowledgeable agent may perform whilst solving a problem, as explained in section 2.4.3. For instance, the number theory in mathematics uses numbers as symbols (e.g., 1.47 or 7.4^3 and rules like the commutative property of addition (i.e., $2 + 4 = 4 + 2$). When analysing a problem at the knowledge level one considers what knowledge concepts are present or lacking when solving the problem. Another question is how the available knowledge concepts couple the actions of an agent to the task being solved, and how do they relate the individual actions to each other and to their environment?

A careful reader may note that the statements above do not make any commitment to the form of knowledge-level models, nor to their

internal representation. The purpose of this higher tier in problem solving is to abstract knowledge as a competence, which, in turn, makes it independent of the physical representation and symbols [NS76]. This stance towards knowledge was an important impetus for artificial intelligence research in general and for knowledge-based systems in particular. Hence, many different implementations of general knowledge-level models can be found in a range of domains, including design [Ste95].

What properties of knowledge-level models make them beneficial for design? First, it is an agent's knowledge of a specific domain that determines its actions during a problem-solving episode in that domain. Different actions of an agent, which seem otherwise random, share a common ground in the knowledge the agent has and uses. Thus, a knowledge-level model is a kind of indexing or referential framework, to which all agent's actions and decisions are related. Such an index is usually complex, richer than the indexes known, for instance, in libraries. Furthermore, a knowledge model is not a linear structure but an interconnected network of records that in addition to determining the agent's actions also define their meaning. Thus, knowledge endows the agent's actions with a specific meaning, purpose and context – with *semantics*.

Knowledge-level models can be shared and transferred among multiple implementations. This aspect is typically illustrated using an analogy with computers: Hardware (i.e., the processor, storage units, memory units) is unique for every computer – it is a symbolic framework; software (i.e., applications and data) is more flexible and, to certain extent, can be shared between different computers. When we replace 'computer' with a generic term 'agent', knowledge-level models become the layer in information processing, on which the collaboration between agents and knowledge sharing take place. To share knowledge the agents must subscribe to a common model, so that they assign the same semantics to interpreting concepts from the shared or transferred knowledge model.

One particular approach to modelling at the knowledge level is based on the entity known as a *common ontology*. Ontology as scientific theory is known to philosophers for centuries; it denotes a theory concerned with the nature and relations of 'being'. This philosophical term was brought to the computer science by Tom Gruber who understands ontology as "*an explicit specification of conceptual repre-*

sentation" [Gru93]. According to this definition, *in knowledge-based systems what exists is only what can be represented* [Gru93]. Ontology is essentially "*a representation vocabulary specialised to some domain or problem matter*" [CJB99]. An agent can use such a vocabulary to express and represent its knowledge in connection with a particular problem or task.

An ontology is an explicit account of what is often referred to as domain theory or background knowledge. An ontology may be developed for different purposes: For instance, for a consistent naming of symbols manipulated in the domain or for a definition of important relations among objects. With ontology one does not define the implementations and the individual bits and pieces of the domain objects, but rather assigns *meanings* to them. The only commitment made by ontology is to the interpretation of objects and relations; ideally, no commitment is made to the agent's internal representations. On the contrary, the commitments an agent makes using a particular ontology may have many internal (or symbol-level) representations. An ontology is thus inherently a knowledge interchange framework...

If multiple agents (including mixture of human and artificial) subscribe to the same commitments in a particular domain, these commitments serve as a common or shared ontology. Such an ontology provides a vocabulary (similarly as a human language does) for the exchange of information. At the same time, ontology also allows assertions to be shared. For instance, data chunk "39°C" yields little immediate knowledge. It may be a measured temperature with different implications in different domains: In weather forecasting it may imply a scorching day; in medicine, a serious fever. Thus, the full potential of a data chunk is revealed only in connection to other data and knowledge in the domain. In other words, it must be interpreted in the domain ontology; only then, more complex assertions and implications may be made.

Some knowledge has a more generic nature than other, and, therefore, may be used in different problems. In other cases it is narrowly bound to a particular problem or task. For example, the main principles of diagnostics are common whether one works as a medical practitioner, a computer support specialist or a shop-floor technician. Terms "cause", "effect" or "location" are generic and form an upper-level ontology that is applicable to generic diagnostics tasks. They may, however, be 'tuned' for the individual domains: For instance,

doctors talk about "symptoms" as a specific type of "effect", and distinguish specific "location" types, such as "upper arm" or "digestive system". The same generic terms may be specialized differently in the device diagnostics, e.g., "power distribution module", "mechanical apparatus" or "input/output system". The specific terms may be grouped into lower-level or domain ontologies. The relations between upper-level and domain ontologies are typically hierarchical.

One may also distinguish domain- from task-oriented ontologies that can be easier to re-use and share than specialized problem-oriented ontologies [CJB99, Mot97, MZ98]. In the above definitions, I mentioned that multiple agents could share an ontology as an interchange model. However, sharing can be understood differently according to two basic paradigms: A common ontology may be shared *spatially*, i.e., there are multiple, physical agents using the same ontology to mediate their knowledge exchange. Alternatively, an ontology may be shared *temporally*, i.e., an agent can re-commit to certain ontological commitments made in the past. In general, sharing may occur along both paradigms – an agent may use knowledge acquired and published by a different agent in the past. Sharing on the spatial paradigm is known as (proper) *knowledge sharing* [Mot97]; sharing on the temporal basis is then called *knowledge re-use*. Nonetheless, both terms imply that an ontology can be shared by multiple agents and problems, or may be re-used in different domains or tasks.

4.2.2. Applications of modelling in design

Similarly as it was the case with the review of current trends in design support (section 4.1), the works mentioned in this section are not exclusive. Due to a vast amount of work done, it is virtually impossible to review the whole spectrum of approaches. Therefore, I give an overview of two large-scale efforts modelling generic design process. It includes the Generic Task Model of Design as a representative of procedure- or process-oriented application of knowledge modelling, and the General Design Theory as a representative of a declarative application of knowledge models in design.

4.2.2.1. Generic Task Model of Design

The key idea behind generic task models is that there are some problem-independent features shared by a class of tasks. For example,

regardless of whether a diagnosis is concerned with health, machine maintenance or pupils' performance, certain actions are repeated and can be observed in all domains. Therefore, it is reasonable to formalise this domain-independent pattern on an abstract level, known as task or task-description level. Knowledge models at this level are then known as *generic task models* (Tansley and Hayball 1993). From a large base of generic task models (or GTM) developed for various tasks, this section briefly describes the GTM for design (GTMD) [BvLTW96, BvLT+96, BvLT98].

GTMD assumes that the knowledge of initial requirements is available at the beginning together with the knowledge of design objects and strategies. Requirements may be ordered using a preferential measure, some of them may be necessary and others optional. GTMD models the preferences in a module named 'requirement qualification set manipulation' (see Figure 4.3). The same artefact may be described from several different viewpoints – GTMD takes this variance in account in the specific module named 'design object description manipulation'. The description of a design object (artefact) is mostly incomplete, and it is stepwise extended during the design process along with the requirement qualifications. The third module in Figure 4.3, 'design process co-ordination', co-ordinates the design process by evaluating its current state and further steps allowed by the knowledge base. The selected strategy then influences the initial requirement qualifications set, as well as the descriptions of design solutions.

In Figure 4.3 each module is decomposed into simpler operations performed during the design process. For instance, for the requirement manipulation module, four tasks are defined:

1. (Requirement set) modification – to determines the strategy for the selection of requirement qualifications; what preferences, what level of strictness are applied to the selected relevant requirements, and so on;
2. Deductive refinement – to infer what requirements are to be considered having decided on certain preferences earlier, and what possible modifications to the current set are admissible;
3. Update of current description – in this step, the selected requirement modifications are enacted and included in the set of requirement descriptions;
4. Update of history – to keep track of the development and stepwise refinement of design

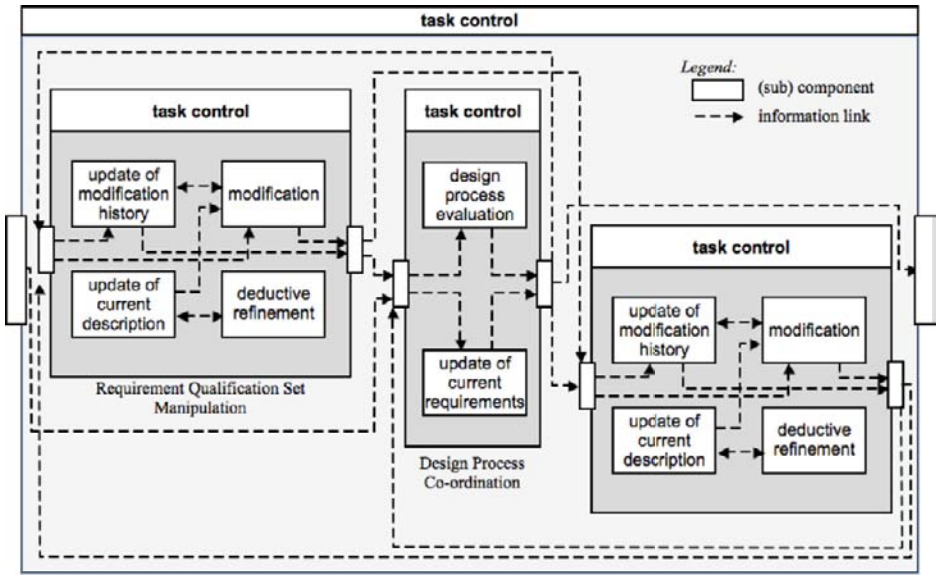


Figure 4.3. Generic task model of design (based on [BvLT+96]).

A similar decomposition is defined for the design object description module; i.e., the module responsible for the development of design solutions: First, candidate objects for the modification are identified, and their changes and modifications are deduced. Next, the changes are incorporated into the existing description, and to keep track of the changes, the history record is updated. The update processes for both modules are related; i.e., a change performed in the requirement module may have an impact on the solution module.

Between these two modules that manipulate the knowledge of design objects, the third module works with the knowledge of design actions. This co-ordinating module assesses the current state of design and suggests amendments in design requirements. In other words, it makes sense to continue with further manipulations of either requirements or solutions, if there is a benefit observable in the development history; for instance, the performance of the design solutions is improving a particular criterion (e.g., a price). This means the design process is progressing (not stuck in a deadlock), and there is still an unused potential in the active strategy.

Design task model in Figure 4.3 is sufficiently abstract to be considered generic. It may be adapted and refined for specific applications or problems. For instance, GTMD was deployed for the elevator

design in Sisyphus-2 domain, which is a benchmarking case for knowledge modelling in design [Yos92]. To account for the specifics of elevator design, the sub-task of a design description (solution) modification is refined into steps like analysis, determination of the modifications, and their implementation [BvLT+96]. The determination sub-task is further decomposed according to the method used for problem solving, such as 'extend-and-revise', 'propose-and-revise', etc. [BvLT+96]

The case study of an elevator design shows how GTMD can be refined for a specific problem, domain, and a suitable problem-solving method. The same case study and similar refinements of GTMD into task-oriented, domain-oriented, problem-oriented and method-oriented appears in [MZ98]. Here, the advantage of generic models is that they can be used independently of each other, and mutually combined as needed. For instance, one may re-use a task model of the 'propose-and-revise' problem-solving method without any commitment to the domain or problem-specific concepts. Such a universal re-usability is compelling, and it was a popular research topic in the 1990-s [Mot97].

GTMD is useful in the design of artefacts of a certain type. If the various combinations of requirements, fixes, preferences, and parameters are well defined and known in advance, then GTMD performs well and reliably. The fewer irregularities there are in a particular domain, the less work is required to tune GTM into a working procedure. However, if the fixes or preferences are ambiguous or incomplete, and the structure of the design process is dynamic, it requires more effort to adapt GTM to such an ill-structured problem. Nonetheless, it is still possible to use GTMD as a template for rapid development of the core application, and extend this core instead of building the whole application from scratch.

Another useful feature of GTMD is its decomposable structure that allows re-using the modules and replacing generic tasks with finer-grained components. Such modularity enhances the re-usability of GTMD components and any refinements of GTMD can be seen as replacements of the generic 'templates' with domain-, problem-, or method-specific plug-ins.

4.2.2.2. General Design Theory

The General Design Theory (GDT) can be traced to early 1980s when it was proposed in Japan. My source of information is [Tom94], where

the theory and its features are evaluated. GDT is a knowledge-level theory interested in the objects that are conceptually manipulated during the design process. GDT regards design as a mapping from the function space to an attribute space, and associates entities from the world to abstract entity concepts. It assumes that all real entities can be described by a finite set of attributes, and that each real entity corresponds exactly to one abstract entity concept. This is rather idealistic, as it suggests that everything one needs to be known about the design problem is explicitly known.

As we already learned, the Closed World Assumption (whereby all objects are well defined) is useful for tackling incompleteness, but if this were the case and a complete knowledge base could be constructed for design, it would deny the need for exploration and interpretation. In the real world, design is a stepwise refinement of design solutions. To comply with such iterations, GDT refers to physical laws as a boundary that constrains the compatibility and applicability of various entities and entity concepts. The iterative development of design solutions to the incompletely described problems is tackled in GDT by the introduction of a meta-model. This is defined as a sub-set of those attributes and values that comply with the physical laws of the real world. In other words, meta-model in GDT is a physically achievable approximation of an ideal design solution. It only considers the objects not contradicting explicit but generic constraints. As new attributes are added to the meta-model to satisfy the constraints, the model becomes increasingly more specific, and finally evolves into a design solution.

GDT distinguishes two kinds of knowledge in design: the knowledge of design objects and the knowledge of design actions. At the object level GDT defines several operators, including deduction, abduction, and circumscription. The operation of circumscription [McC80] allows to 're-form' the knowledge base to resolve a constraint violation. However, this is also a shortcoming of the GDT model, since circumscription only *restricts* the set of facts (see section 3.2). It relies on the rule that the objects mentioned as having certain property (e.g., compliance with a constraint) are the only objects needed (to satisfy a law or a constraint). Thus, circumscription in GDT can restrict the choice of the attributes and their values to restore the consistency of the solution model. Consequently, GDT is unable to come up with a proposal of a solution extension that may violate some

constraints of the physical laws. Thus, the occurrences of unexpected behaviours [Sch83] or physical contradictions [Alt84] are 'blocked' by circumscribing the problem solving theory.

According to GDT, design actions are modelled as a deductive mechanism comprising a set of rules determining what to do with the object level. Apart from allowing heuristic predictions, GDT does not recognise that the action knowledge may be occasionally tacit. Thus, the model does not fit the situations, in which, designers approach the problems opportunistically and decide on their strategies on-the-fly, and in which the strategic knowledge involves complex reflective turns rather than deductions [Can98]. Yet GDT supports multiple design contexts and maintains the consistency within a context through a truth maintenance system. It acknowledges that design is a process of solution refinement that rarely begins with complete initial requirements. Finally, it agrees that design is neither synthesis nor analysis but a combination of both.

On the other hand, the 'talkback' from the space of solutions back to the space of requirements is limited. The only form of feedback in GDT is the violation of constraints by a solution model that can be resolved by circumscription. The GDT model, although being iterative, is a kind of spiral rather than oscillation between the functions, the attributes and their values. In GDT new knowledge may be articulated by a designer, but the model does not support this articulation. It does not propose any models to associate new constraints with any reflection or tacit assessment. The designer is 'a black box' doing the evaluation and proposing modifications, and it is not the aim of GDT to investigate the details. The theory keeps focused on investigating the space of the physically admissible designs, and on maintaining logical consistency in the development of solutions. Some more recent extensions of GDT can be found, e.g., in [TN94].

4.3. Summary or the role of modelling in design

One of the major problems with design is its initial under-specification. In addition to developing a design solution, one has to refine his understanding of a particular design problem. The formulation of conceptual frames for a particular problem was highlighted as helpful in seeing the current problem in a more familiar light [Sch83]. Such a conceptual commitment depends on the use of ontologically equiva-

lent conceptual vocabulary for the problem interpretation.

The capability of 'seeing as' involves sharing ontological commitments between the design cases. Certain knowledge-level entities can be transferred from the previous design case to the current one. One may even argue that, in respect to those particular commitments, the previous case is a model for the current design problem. The previous case with its conceptual vocabularies, design specifications, as well as design solutions, is easier to work with than the incomplete current case. The previous problem might not be simpler in terms of structures or behaviours. Nonetheless, it is *explicitly described* and provides *means for 'completing' the underspecified design situations.*

Thus, knowledge models in design are likely to have a greater role in the problem specification than it is presented in the existing implementations. To make use of the full potential of conceptual re-use, it is desirable to use problem-specific languages for the interpretation of the individual design problems [Fis92, Goe95]. To make sharing and re-use feasible, it is also desirable to cross-reference these interpretations to a shared index; for example, in the form of a knowledge model that contains problem-independent commitments. An example of such a model is the ontology of components, substances and actions [BC85] or the ontology of physical quantities and laws [Bor97, Wev99].

4.4. Gaps between the theory and practice

As it is visible from my review, the positions on the issues like design under-specification are split, and there are contradictory views on the subject. This controversy arises because many views do not attend to design holistically. First, I ascribed an important role to the exploration in design. Nevertheless, despite the fact that many implementations accept the importance of problem interpretation by exploring past solutions, there is not enough known about what is the place of the tacit or hard-articulate knowledge in modelling such an interpretation.

Similarly, if we restrict our attention to the development of design solutions, it is sufficient to assess the solution admissibility against the given requirements and constraints. Nevertheless, such assessment is difficult when we intertwine the synthesis of design solutions with their analyses, in order to 'complete' the underspecified problem. What if we want to change the current problem specification to

re-formulate the design task and test a particular perspective? How is the success or failure of such an amendment assessed?

From the reviewed literature we learned that the designers often use their deep knowledge and experience to re-formulate the current problem interpretations. However, the conceptual and operational models do not formally incorporate these empirically observed facts. The formulation of frames and exploratory probes, as well as their re-formulation is opportunistic rather than well understood. There is an empirical evidence of the different types of exploratory probes be used for the problem (re-)interpretation. Such probes are difficult to express explicitly, but this does not mean that they do not exist or do not deserve a place in formal models of design.

In real design, the trick of the trade is not in creating new structures but rather in 'uncovering' of known structures and bringing them in focus. Cognitive studies refer to insight, or a designer's deep domain knowledge, as a means to recognise such hidden relationships in the design problem [Boy89, PL88]. This knowledge is also responsible for informing the strategic decisions; i.e., how to continue with the design [BvLT98, Can98]. However, how does insight work at the level of knowledge? Insightful acts are difficult to capture in formal representations, but it is an interesting challenge to locate their form and place in the design process.

As many reviewed approaches show, it is the capability to draw analogies in and across domains that helps to overcome contradictions emerging in a particular perspective. The process revealing and resolving such contradictions in explicit terms of the current or amended problem frame is the key to understanding insight. To resolve contradictions in the existing problem, the design problem itself (i.e., its frame) has to be perceived in different conceptual terms. Designers impose a kind of frame to attend to an ill-structured and under-specified design situation. Thus, it is the process of framing and re-framing that deserves a place in the formal model of the design process. The reason why this topic is interesting is my belief that *framing is related to knowledge-level modelling*. One may see a conceptual design frame as a vocabulary (based on the vocabularies of the past design tasks that are familiar and much better structured), which can be used for the explicit articulation of one's understanding of the under-specified problem.

In the next part, I define the *design frame* formally in terms of

knowledge being used and manipulated, and associate this term with the interplay of the problem specification and the construction of a solution. In chapter 5, an initial version of the Recursive Model of Framing in Design is given.

Part II

Recursive Model of Framing in Design

This page intentionally left blank

Chapter 5

Theory of Framing in Design

The notions of 'design frame' and 'design framing' are not new in the design research; they can be seen in [Sch83], where the author talks about shaping of design situations. Similar notions appear also in [Fis92, KW96, SCD+90], where authors talk about contexts, views and perspectives. Nevertheless, the gap lies in the interpretation and the formal clarification of these terms. Problem framing is an important operation that precedes the problem solving and complements it, but what are the implications of framing at the knowledge level?

Another motivation for my formal model has roots in the work of software engineering community [Fis92, NE00]. The conceptual duality of design specification and design solution, and their equal importance were recognized in [SB82]. However, the relationship of these two distinct conceptual spaces is not investigated in depth [Nus01]. What is happening with the designer's knowledge during problem framing? Can this operation be expressed in a formally or semi-formally?

5.1. Essential definitions

The closest attempt in the literature to formally address the issue of problem shaping can be found in the notion of *a design perspective* that is defined as "a point of view, which implies that certain design goals exist, certain bodies of design knowledge are relevant, and certain solution forms are preferred" [NSH94]. Design perspective expresses the designer's intentions; it is a kind of terminological

vocabulary used in the problem solving phase. This foundation is helpful, and I will base my definitions on the above-given position.

5.1.1. Basic concepts for model of framing in design

It seems to be obvious to say that designers solve *design problems*; let us denote a design problem as $\mathscr{DP}$. This problem is usually vague or ill structured, and to solve design problem $\mathscr{DP}$, the designer has to characterise it by an explicit selection and application of suitable elements from a space of applicable problem specification primitives $\mathscr{S}$. Such explicit problem specification, denoted as $S \subset \mathscr{S}^*$, provides a context for choosing the design methods and domain theories. These methods and theories are tools enabling the designer to satisfy the explicit specification. However, since problem specification S is only *an interpretation of* $\mathscr{DP}$, the solution satisfying S may not be a solution to $\mathscr{DP}$. This important distinction is discussed more thoroughly later. Now, let me develop the theory of framing step by step.

To solve design problem $\mathscr{DP}$, the designer associates it with a suitable specification S, and tackles the problem specified (interpreted) in this fashion. In other words, the designer attempts to specify the vague design problem $\mathscr{DP}$ in terms of some statements S s/he is familiar with. These statements are formulated and instantiated using the set of selected specification primitives $\mathscr{S}$. We may say that the designer circumscribes [McC80, McC86] the design problem by declaring that only the statements from the explicit specification S are needed to characterise it[1].

Such statements, however, can be only made within certain conceptual boundaries, in *design frame* Φ. I define design frame Φ as a pair of two circumscribed knowledge spaces: the chosen problem specification primitives $\mathscr{S}$ and the chosen problem conceptualisations $\mathscr{T}$. Thus, 'framing a design problem' means articulating a set of conceptual objects $\mathscr{T}$ that will be used to solve the problem, as specifiable using concepts from $\mathscr{S}$ (relevant problem specification primitives). Design frame is denoted by symbol Φ, and can be expressed as: $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$.

[1]This is indeed a kind of circumscription whose purpose is to close the designer's understanding of an incompletely defined, ill-structured problem.

Two additional symbols $\mathscr{S}^*$ and $\mathscr{T}^*$ are needed for a formal description of the above-mentioned circumscribed knowledge spaces. First, $\mathscr{T}^*$ is a closure constituted by applying the selected conceptualisation $\mathscr{T}$ to an appropriate domain theory DT. Domain theory DT is problem-independent knowledge in the sense that it is applicable to different problems. For example, physics of rigid bodies is a domain theory applicable in the design of elevators or shock absorbers. However, for differently conceptualised problems, different parts of DT are applicable. Abstract domain theory DT has to be instantiated for a particular conceptual base $\mathscr{T}$ to obtain a usable vehicle for solving design problem $\mathscr{DP}$. Let me refer to this vehicle, that is, to the closure $\mathscr{T}^*$ as *a problem solving theory.*

Similarly, closure $\mathscr{S}^*$ is a hypothetical instantiation of the problem specification statements that can be articulated in the chosen set of problem specification primitives $\mathscr{S}$ and the selected domain theory DT. Knowledge spaces labelled as $\mathscr{S}^*$ and $\mathscr{T}^*$ are closures and instantiations of finite sets of conceptual primitives $\mathscr{S}$ and $\mathscr{T}$, but they may not be fully enumerated in explicit terms or completely formalised.

Let me interpret the above-definitions in the terms of my review work (chapters 3 and 4):

1. The problem conceptualisation $\mathscr{T}$ corresponds to an ontology, a vocabulary of conceptual objects, which the designer makes available for expressing a statement about a particular design problem and its solutions. This conceptual set includes the terminology for the definition of functional and structural objects, and allows for problem-specific mappings between the functions and structures, various types of behaviours, or relations [BC85, QG96].
2. The domain theory DT may be seen as an upper-level ontological vocabulary that defines the background [WAS95] against which any conceptualisation can be tested. Domain theory per se is too abstract to be of any direct use; to derive a useful problem solving theory, DT must be instantiated by a particular problem conceptualisation (set $\mathscr{T}$ above).
3. The two sets described above define what knowledge is available for problem solving, in a particular design situation. The conceptual basis of the problem (set $\mathscr{T}$) must be interpreted in the light of a relevant domain theory (DT) to allow and drive

design problem solving.

4. Set $\mathscr{S}$ is an ontological vocabulary describing the basic terms and types of relations that the designer can use for the problem specification. It may be defined as a task/method ontology, and at its simplest, it may contain concepts like 'constraints' and 'requirements'. All design problems rely on these two types of specification primitives; some need more details such as 'fixes', 'assumptions', or 'preferences'.
5. A hypothetical space of potential problem specifications $\mathscr{S}^*$ that can be articulated in a given conceptual frame complements problem solving theory $\mathscr{T}^*$. Its principal purpose is to allow the designer to explicitly express his or her intentions in connection with a particular problem. It is a set of statements that can be formulated about the functionality or purpose of the elements in a particular problem solving theory $\mathscr{T}^*$.
6. The set of problem specifications $\mathscr{S}^*$ is an instantiation of the conceptually allowed specification primitives $\mathscr{S}$ with the terms known in a particular problem solving theory $\mathscr{T}^*$. An introduction of a new specification primitive, such as 'an assumption' into $\mathscr{S}^*$, may also change how the objects in $\mathscr{T}^*$ are used, and what methods and operations are carried out to find a candidate design solution $T \subset \mathscr{T}^*$.

As mentioned earlier, a design frame does not exist on its own; it must be constructed on the fly using the information that is believed to be relevant to design problem $\mathscr{DP}$, and that is available as an initial design brief. Typically, a designer uses the design brief to decide on the initial problem specification S and to identify similar and familiar past design situations. Thus, a design frame is a form of *a relation of similarity* with past design cases. This is a desirable feature, because it reflects the empirical observations of "designers seeing one problem as another" (see section 3.3). Design frames are thus subjectively constructed interpretations of a potentially vague and ill-structured design problem $\mathscr{DP}$. Their role is to narrow the space of all possible interpretations to one that is better structured. Such a structured frame informs the selection of applicable domain theories, design methods, tools and so on. The design frame is also a conceptual and rather abstract boundary for tackling design problem $\mathscr{DP}$, and to deploy it in any meaningful way, the individual conceptual sets defined in this section need to be 'linked'. Let me therefore,

define some relationships that apply to the above-defined abstract entities.

5.1.2. Predicates for the model of framing in design

Design starts with an articulation of a conceptual frame Φ that informs both problem specification and solution construction using several knowledge sources enabling this operation: for example, the design brief provided by a customer or the previous knowledge and experience. Using these two sources the designer aims to characterise problem $\mathscr{D}\mathscr{P}$ by explicitly noting the essential features desired from the design solution. Formally, this operation can be represented by an explicit assertion of the fact that a certain set of statements S '*specifies*' problem $\mathscr{D}\mathscr{P}$:

$$\exists S \subset \mathscr{S}^* : specifies_{\Phi}(S, \mathscr{D}\mathscr{P}) \tag{5.1}$$

Design continues with an attempt to formulate a minimal set $T \subset \mathscr{T}^*$ that '*satisfies*' a given problem specification. However, before any solution can be proposed, the other 'half' of the design frame, set $\mathscr{T}$, needs to be articulated explicitly to give a vocabulary for the construction of design solutions. The conceptual vocabulary is decided by the designer taking into account the initial problem specification S (see equation 5.1); it is chosen so that it bootstraps the problem solving theory for the given problem specification. The 'given problem specification' is in my theory understood as set $S \subset \mathscr{S}^*$, and is defined as a formulation of explicit design requirements and/or constraints (see also equation 5.3). Thus, S always contains only the statements characterising design problem $\mathscr{D}\mathscr{P}$, to which the designer made an explicit commitment.

Having decided on the conceptual foundation $\mathscr{T}$,the next step comprises the articulation of a problem solving theory $\mathscr{T}^*$, from which a design solution can be constructed. Once design problem $\mathscr{D}\mathscr{P}$ was interpreted in explicit terms of a particular problem specification $S \subset \mathscr{S}^*$ and the problem solving theory $\mathscr{T}^*$, a minimal sub-set of the problem solving theory is selected to deliver the explicitly specified requirements. In other words, the designer tries to shrink the space defined by problem solving theory $\mathscr{T}^*$ into a manageable chunk that can be manipulated, explored or searched. This manageable chunk corresponds to term '*solution model*' reviewed in [Alt84, Ger90]. Due

to its generative nature, however, I will refer to formula $T \subset \mathscr{T}^*$ as *a problem solving model*.

Formally, a problem solving model is a minimal sub-set of the problem solving theory that satisfies the explicit problem specification. Telation '*satisfies*' is binary, because it associates a problem solving model T with the explicit problem specification S (using design frame Φ as a restriction of the contextual validity of the relation) – see equation 5.2. The existence of a problem solving model T inside problem solving theory $\mathscr{T}^*$ '*proves*' that the commitment to a particular conceptualisation $\mathscr{T}$ was correct.

$$\exists T \subset \mathscr{T}^* : satisfies_{\Phi}(T, S) \wedge (\neg \exists Y \subset T : satisfies_{\Phi}(Y, S) \quad (5.2)$$

One possible definition of relation '*satisfies*' takes into account semantic distinctions in the explicit problem specification: It is possible to distinguish design requirements R from design constraints C, and proclaim the problem specification as the union of the two, $(S = R \cup C)$ [WAS95]. Requirements are statements demanding the explicit presence of a particular feature, whereas constraints are the conditions that must not be explicitly violated by a design solution. More on this conceptual distinction appears in section 5.3.4, but at this stage, a simplified version in equation 5.3 suffices as an operational definition of relation '*satisfies*':

$$satisfies_{\Phi}(T, S) \iff ((S = R \cup C) \wedge (T \models R) \wedge \neg (T, C \vdash \bot)) \quad (5.3)$$

Symbols in formula 5.3 have their usual meaning [GN87, Lev74, Men79]: '$\models$' denotes semantic entailment, '$\vdash$' denotes a proof-logical implication, and '$\bot$' stands for an empty formula, the logical contradiction. Thus, the translation of this definition is: "Theory T is a problem solving model in respect to a given explicit problem specification S and design frame Φ, if it is *complete* in respect to the required features $(\forall r \in R : T \models r)$, and simultaneously it is *admissible* in respect to all constraining conditions $(\neg \exists c \in C : T, c \vdash \bot)$." In other words, the problem solving theory must deliver all desired features without breaking any constraints. Admissibility and completeness are both explicit categories used in the computation of relation '*satisfies*'. Details of how an admissibility of any candidate solution may be assessed are in section 5.3.

Recall that explicit problem specification S is an explicit interpretation of design problem $\mathscr{D}\mathscr{P}$, which is used for problem solving. Thus, the existence of a problem solving model T, for which the relation of explicit satisfiability, $satisfies(T, S)$ holds, is a necessary but not sufficient condition for declaring T *a design solution*. In addition to satisfying the explicit problem specification, the discovered problem solving model T must be also acceptable as addressing the original problem $\mathscr{D}\mathscr{P}$. The irony is that it is not possible to define such a relation as $acceptable_{\mathscr{D}\mathscr{P}}(T)$ in explicit terms and in advance due to the fact that acceptability is appreciated subjectively and tacitly, and it cannot be fully expressed in the same languages as the conceptual sets $\mathscr{S}$ or $\mathscr{T}$ forming the design frame. Nevertheless, such a relation may be defined as 'a residual', as in formula 5.4:

$$satisfies_{\Phi}(T, S) \wedge \neg acceptable_{\mathscr{D}\mathscr{P}}(T) \implies \neg specifies(S, \mathscr{D}\mathscr{P}) \tag{5.4}$$

What does it mean that $acceptable_{\mathscr{D}\mathscr{P}}(T)$ is a residual relation? It means the same as the statement from section 2.3 saying that certain tacit decisions cannot be stripped of their contextual background. It may be difficult to define exact conditions of acceptability in advance and evaluate them, but a designer may proclaim a certain problem solving model acceptable or not when s/he *reflects* on it. The tacit decision of acceptability makes sense only in a particular context – in formula 5.4 the context is represented by design frame $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$, where $S \subset \mathscr{S}^*$ and $\mathscr{S}^*$ is a closure over the set of problem specification primitives $\mathscr{S}$.

Another way to translate formula 5.4 is that whenever an otherwise sound problem solving model T is not accepted by the designer as a design solution, it suggests an incorrect or incomplete specification of design problem $\mathscr{D}\mathscr{P}$. The designer may not be able to articulate which criterion is violated by the candidate solution; s/he does not like it, for example, because the candidate solution delivers unexpected results or exhibits undesired behaviours. We may also say that the interpretation of design problem $\mathscr{D}\mathscr{P}$ in the explicit terms of specification S does not fully reflect the gist of design problem $\mathscr{D}\mathscr{P}$, and as such it must be amended.

Let me now turn to the core of my model – the recursive repetition of problem framing, acceptability assessment and problem re-framing, and define it as a sequence of the conceptual predicates defined earlier.

5.2. Recursive model of framing in design

The building blocks of the Recursive Model of Framing in Design are the conceptual entities identified in the previous section, in particular:

1. problem solving theory $\mathscr{T}^*$,
2. explicit problem specification S,
3. problem solving model T, and
4. conceptual design frame Φ

The model is defined as a sequence of decisions based on the three predicates '*acceptable*', '*satisfies*' and '*specifies*' from section 5.1. The sequence runs across several mutually dependent levels, where Level 0 denotes the fundamental decision in the sequence, the initial specification of design problem $\mathscr{DP}$ in the explicit terms. Following this initialisation, the model continues with a *recursive* loop of the three predicates; with the acceptability assessment serving as a stop condition.

The model treats design as an interplay of two distinct knowledge-level actions represented by predicates '*specifies*' and '*satisfies*'. The former predicate is amending the problem specification, and the latter attempts to solve that specification. These two actions have a potential to generate and change the explicit design knowledge. To enable this step-by-step knowledge generation, an auxiliary predicate is needed to facilitate the decisions about further steps in the design process. This auxiliary definition is useful, so that the model can be read intuitively as a sequence of decisions followed by actions.

In addition to the mentioned decision on the acceptability of the design solution, there is only one other decision defined in the model. This decision complies with the reviewed literature, in particular, with Schön's theory of reflection in-action (section 3.3). Its purpose is to construct an exploratory probe in an attempt to modify the current design frame. Thus, a frame can be modified if it is possible to explicitly articulate new sets $\mathscr{T}_N$ and $\mathscr{S}_N$ that provide a different vocabulary for attending to design problem $\mathscr{DP}$. The frame modification is formally presented in formula 5.5.

$$modifyFrame(\Phi) \iff \exists \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle : (S' \subset \mathscr{S}_N^*) \wedge \\ \wedge specifies_{\Phi_{NEW}}(S', \mathscr{DP}) \tag{5.5}$$

Predicate '*modifyFrame*' decides on whether to continue with the design or not, based on the existence of a new conceptual frame that can be applied to the ill-structured design problem $\mathscr{DP}$. In the definition, the new frame is expressed as a new pair: $\Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle$. The new, modified or shifted frame Φ_{NEW} is not selected randomly, but so that design problem $\mathscr{DP}$ can be differently but still meaningfully characterised in this amended frame. As the frame changed, its vocabularies for the specification of the design problem change as well; this is expressed in the definition by symbol S' in predicate $specifies(S', \mathscr{DP})$.

The decision about re-framing can be made in a specific phase of tackling design problem $\mathscr{DP}$, and it has one important and unusual feature. Namely, it is not merely a "yes/no" decision; it is also a generative operation that is used to improve the designer's understanding of the design problem. The careful reader may note that predicate $modifyFrame(\Phi)$ affects only the specification of an incompletely given design problem, and stands for generating new knowledge about how the problem may be specified differently. This is an important side effect of the definition that emphasises the recursive nature of the model, and features the interaction between the problem specification and problem solving as argued in section 3.4.

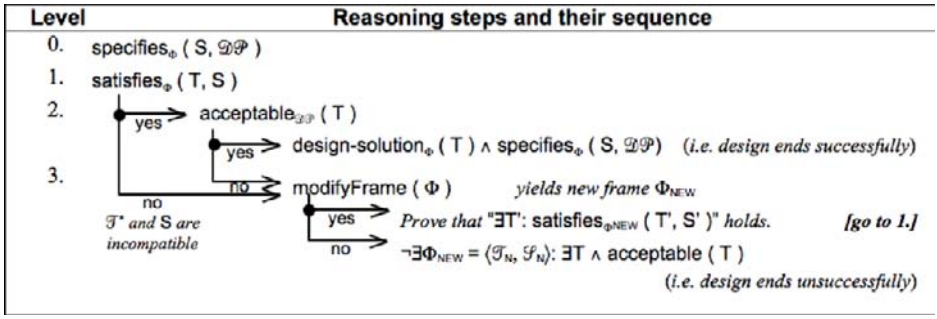


Figure 5.1. Recursive model of framing in design as an interaction between problem specification and problem solving

The schematic representation of the model in figure 5.1 shows that a positive outcome of the decision contemplating the frame modification is directly followed by a potential change in the solution space. This potential change is represented by predicate '*satisfies*' that manipulates the problem solving theories and models. Thus, the men-

tioned interplay of the two knowledge spaces is observable in the exchange of information and control during the design process. Frame Φ enabling a particular explicit problem specification S is thus verified by an attempt to find a satisfactory solution to such a specification. If at least one explicitly admissible solution T is found (that is, a problem solving model satisfying the explicit problem specification exists), it must be assessed for its acceptability.

Provided that the acceptability check is successful, this means the designer is satisfied with a particular solution proposal and its completeness. In other words, it is now possible to declare it a design solution to the under-specified design problem $\mathscr{DP}$. If the designer does not accept the candidate solution as addressing the problem, frame modification is triggered (Level 3 in figure 5.1). In this operation, the designer reflects on the inadequacies of the design frame Φ and its constituting sets, and articulates a new frame Φ_{NEW} as a probe to rectify them.

If another frame can be found to help the designer to update the vocabulary for the problem specification, and to introduce new requirements or constraints, the exploratory probe is confirmed. The next step comprises the validation of the exploratory probe in terms of finding a new solution candidate T' to address the amended explicit understanding of the problem S'. In other words, the validation incorporates the reflection-in-action; the designer's inarticulate feelings about the solution unacceptability are validated in the construction of a new admissible design solution. Simultaneously, the model of the design process loops to Level 1, and the sequence repeats in a recursive manner.

In addition to the reflective exploration, the model also accounts for a specific case when it is not possible to satisfy a particular specification using the selected problem solving theory. This case is reminiscent of the scenario from section 3.1.4, where the designer discovered an explicit contradiction in the problem specification when he adopted a certain frame containing familiar conceptual objects for a solution construction. Using the language of my model, that design frame consists of two mutually incompatible knowledge spaces $\mathscr{T}$ and $\mathscr{S}$. If the incompatibility is explicitly provable, this can also trigger design problem re-framing.

The breakdown in Figure 5.1 is developed to address the theoretical assumptions that came out of the literature review. The sequence

of decisions would then obey a few simple rules that circumscribe the category of design problems that can be modelled in this way:

1. When using terms 'requirements' and 'constraints', we always mean hard, strict demands that must not be relaxed; see [Fox94] for constraint relaxation techniques.
2. In this model, I do not consider the design problems, where a problem specification can be simplified or otherwise relaxed; these issues are covered by other research [ZF94].
3. Problem specification may be changed by a monotonic extension of the initial sets S and $\mathscr{S}$; new, refining statements about the specification can be made under specific circumstances.
4. The monotonic extension of a problem specification corresponds to a designer's attempt to fine tune the problem solving model and to narrow the number of derivable alternatives.
5. Sentence in the form "*Prove that (...) holds ...*" represents a recursive step starting always at Level 1 of the recursive model.
6. The recursion represents a designer's attempt to address design problem $\mathscr{DP}$ by a modification to the available knowledge sources. It can also be understood as an order to an agent to evaluate a particular predicate with new arguments provided.

Next, the recursive model is explained by identifying three 'sub-models' subsumed in the RFD model. These are, in fact, generic schemas of different types of reasoning that can be observed in a design process. They are intended to emphasise the mutual dependence of different knowledge sources, as well as the interaction of explicit reasoning strategies with those using a designer's inarticulate knowledge. Particularly important for my analysis of problem framing are sub-models 2 and 3 with their non-monotonic introduction of new knowledge into the design process. Predicate chains in the schema definitions reflect the paths between the levels in Figure 5.1.

Reasoning schema 1

$$satisfies_{\Phi}(T,S) \underset{\text{yes}}{\longrightarrow} acceptable_{\mathscr{DP}}(T) \underset{\text{yes}}{\longrightarrow} designSolution_{\Phi}(T)$$

This is the most trivial design model from the perspective of interaction between articulate and inarticulate reasoning. It represents a scenario when the selected design frame Φ contains a suitable problem solving theory $\mathscr{T}^*$ and a sound problem solving model T, which

can be inferred from theory $\mathscr{T}^*$. Furthermore, there is little need to distinguish between admissibility and acceptability of the inferred problem solving model, as the admissibility criteria are well defined, and a solution is acceptable whenever it is admissible and complete.

This schema typically accounts for well-defined problems or parts of the ill-structured problems that are well defined. This is a class of problems, where design requirements, constraints, fixes, and elements are fully known in any particular design step. No tacit knowledge or frame amendments are needed, and the problem can be solved using generic or dedicated algorithms. The essence of this model can be expressed as: "*I know how to define the problem, and I have all the necessary information available.*"

Reasoning schema 2

$$satisfies_{\Phi}(T,S) \underset{\text{yes}}{\longrightarrow} acceptable_{\mathscr{DP}}(T) \underset{\text{no}}{\longrightarrow} modifyFrame(\Phi) \rightarrow \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle \underset{\text{yes}}{\longrightarrow} satisfies_{\Phi NEW}(T',S')?$$

This schema tackles the situation when there is at least one logically sound solution available but the designer does not accept it. This situation has its origins in the difference between the explicit design frame $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$, which is used for the problem specification and solution construction, and the designer's expectations. Chosen conceptual sets $\mathscr{T}$ and $\mathscr{S}$ do not contain the vocabulary for expressing those expectations explicitly. Moreover, the designer often becomes aware of these expectations only when s/he proposes some candidate solutions. In other words, the logically sound solution is subjected to a tacit assessment, and during this assessment a new feature emerges that was unaccounted for in the original frame Φ.

The designer reflects on the proposed solution, the explicit problem specification and tries to pinpoint the part of the problem s/he is dissatisfied with. During the reflection a synchronisation of the explicit but incomplete conceptual frame and the tacit expectations is performed. The process of uncovering tacit expectations typically requires a construction of exploratory probes to test different amendments to the original frame. In general, each probe may yield a new frame Φ_{NEW}, which can be used to interpret design problem $\mathscr{DP}$ differently than the original frame Φ.

The scenario covered by this model is distinct from that described in schema 1. The main difference is in the fact that new requirements,

constraints or other conceptual objects can be formulated only after the design frame shift. Without a frame amendment the requirements remain hidden and tacit; in the original frame Φ there are no conceptual means to express such tacit expectations explicitly, but new means can be obtained in a new frame Φ_{NEW}. Verbal summary of this class of problems is: "*I cannot solve the problem in this way, something is missing in my approach. What if I took a different view (...and assumed...)?*"

Reasoning schema 3

$$satisfies_{\Phi}(T,S) \underset{\text{no}}{\longrightarrow} modifyFrame(\Phi) \rightarrow \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle \underset{\text{yes}}{\longrightarrow} satisfies_{\Phi NEW}(T',S')?$$

This is a situation, in which the chosen frame Φ contains a problem solving theory $\mathscr{T}^*$ and an explicit specification $S \subset \mathscr{S}^*$ that are mutually inconsistent. In other words, there is a logical contradiction when using the axioms of the deployed domain theory DT conceptualised by $\mathscr{T}$ to satisfy problem specification S. Consequently, no sound problem solving model and design solution can be found in such an unsound theory. After changing the frame from the original Φ to new Φ_{NEW}, new conceptual objects may emerge in set $\mathscr{T}_N$) or the problem specification may be re-visited ($\mathscr{S}_N$). In the new frame Φ_{NEW} the conflicting axioms (usually constraints) of the original used domain theory DT are interpreted differently. The new interpretation of an axiom is a non-monotonic operation influencing the selection of domain theories – new domain theory DT_{NEW} is then created to suit the amended conceptual frame better than the original one that led to contradictions.

The class of problems addressed by this model is relatively well structured. The designer has explicit knowledge of a logical contradiction in the problem solving theory. There is an explicit (typically constraining) condition that embodies the contradiction formulated in the vocabulary of the original frame Φ. This helps to search for a new frame that can be assessed for compliance with this problematic constraint. As in schema 2, the designer does not have the expressive means in the original frame to remove the contradiction; he needs a new frame to expose it. This model is thus close to the research on the resolution of physical contradictions in the inventive problems

(sections 3.1.4 and 4.1.4). The following statement can describe the class of such problems: "*I know exactly what is wrong, and I know of some options that may fix it.*"

The schemas following directly from the RFD model described above are defined at an abstract level of concepts and conceptual design frames. They are not computational or operational models, and their purpose is to model common reasoning strategies that underpin a designer's decisions at the knowledge level. However, it is also interesting to investigate how problem specification may be satisfied, how different solutions can be generated from particular problem solving theories, and how to discover a suitable familiar representation for framing the design problem. Some of these topics are re-visited in section 5.3, where one operational model for predicate '*satisfies*' is presented. My main focus, however, is to develop a conceptual model and validate it empirically. Hence, the analysis of my empirical observations in chapters 7 and 8 will aim to give support to the three schemas from this section.

5.3. Operational models of operator `satisfies(T, S)`

Let me propose one possible operational model of how relation '*satisfies*' can be calculated and evaluated. This is only an overview of a few techniques refining the conceptual level of the RFD model at a lower, computational level. In this section I illustrate what I meant in the previous sections by "a designer searching for a sub-set of the problem solving theory ($T \subset \mathscr{T}^*$) that satisfies explicit problem specification S.

The reasoning during the satisfaction phase will be expressed the language of logic. Logic is a base to other studies and sciences, and is defined as "a study of methods of reasoning" [Men79]. Logic is concerned with the form of reasoning rather than content, and as such it may not be a suitable means to explain all peculiarities of decisions about design. Nevertheless, its high expressive power makes logic a useful language for the explicit representation of the phase, in which explicitly specified design problems are solved. In accordance with the definition of 'knowledge level', the medium of knowledge may be implemented in various symbol-level representations. In the next sections I will use predicate calculus (first order logic) as the

symbol-level representation to express the elementary reasoning steps for modelling the problem satisfaction phase. Nevertheless, what is covered by the logical theory is the reasoning with the explicit parts of design knowledge only.

In the following sections, I touch on three basic operations, a designer may perform from a logical point of view: (i) logical abduction, which is related to the synthesis and divergence in design, (ii) logical deduction, which focuses on the design analysis and the explication of consequences of a given set of premises, and finally, (iii) the truth maintenance and consistency of the problem solving theory is presented as a way to assess soundness and admissibility of design solutions.

5.3.1. Abduction from a logical theory

Abduction is one way of connecting a triplet of logical formulas φ, γ and $\varphi \implies \gamma$. Simply said, knowing that γ and $\varphi \implies \gamma$ are valid formulas, we are allowed to hypothesise that φ may be true as well, because it accounts for observing γ. Abduction is a form of a hypothetical reasoning [Lev89] that seeks formula φ, which together with the background knowledge, would sufficiently account for the valid formula γ. To re-phrase the definition, we say that knowing γ and $\varphi \implies \gamma$ one can hypothesise the validity of φ. Abductive reasoning process runs 'backwards'; from the known consequences it hypothesises possible causes and premises.

The fundamental term in the definition is term 'sufficient'. Consider diagnostics as an illustration of abductive reasoning, when one tries to explain observation γ, while this observation is known to be a consequence of a symptom φ. The observer may not know for sure the exact reason why effect γ occurred, but if he hypothesised symptom φ as its sufficient cause, it would allow explaining effect γ as a consequence of premise φ. Next, one usually looks for a proof of the hypothesised symptom φ in the investigated system. If such a proof of symptom φ is found (say, by measuring it in the system), this also confirms the hypothesis suspecting the symptom being responsible for effect γ. It is indeed, sufficient to account for the observation made but it is not inevitable.

There are different strategies for abduction [Poo89a, dKMR90, Poo89b, Poo90, Rei87]. However, reasoning by abduction is always

a *hypothetical* form of reasoning. It is based on logical theories, but unlike deduction, it does not conclude the inevitable consequences, only the sufficient causes. There is no guarantee that the abduced formula is the real cause of a particular observation. In other words, abduction is not a sound step in any logical proof.

To apply abduction in design, suppose $r_1 \in S$ is a desired functionality of a designed artefact, where S is a set of explicitly specified functions or properties (as defined in section 5.1). Assume that statement r_1 appears in problem solving theory $\mathscr{T}^*$. Also assume that an artefact exists that is represented by a set of formulas $T \subset \mathscr{T}^*$. If in the problem solving theory T delivers functionality r_1 (that is, $T \vdash r_1$) and T is consistent with the explicit problem specification (that is, $S \models T$), then we say that T is abducible from theory $\mathscr{T}^*$. Artefact described by T has a sufficient means to ensure that the requested feature r_1 will occur, as desired. In this sense, we discovered potential solution T to a partial specification $r_1 \in S$ – by a speculative rather than sound logical reasoning.

Nevertheless, precisely of the capability to hypothesise, abduction is a suitable strategy to implement the exploratory design paradigm. This is a corollary of the abduction making tentative choices of further directions based on the current position in the design space. Tentatively abduced formulas may be re-visited later, when new design constraints or requirements emerge, to be confirmed or retracted.

5.3.2. Deduction from a logical theory

Unlike the hypothetical abduction that looks for sufficient means, deduction from a logical theory is the only *sound* operation in the predicate calculus. To deduce means to derive the *inevitable and necessary* conclusions from an arbitrary set of logical theorems and axioms in the logical theory. Provided that certain premises are believed as valid, deduction only propagates this belief throughout the logical theory, and explicitly formulates the theorems that must be also believed. The necessity is based on the assumption of rationality [New82], whereby if an agent believes in a particular axiom, it must necessarily believe in all of its consequences.

Logical deduction starts with formulas α and $\alpha \implies \gamma$, and concludes the validity of formula γ. It is defined as a sequence of well-formulated formulas $\alpha_1, \ldots, \alpha_n$ such that $\alpha = \alpha_1$ and $\gamma = \alpha_n$, where

each α_i is either an axiom in the logical theory, a logical tautology, or a direct consequence of the existing theorems by an allowed rule of inference [Men79]. If such a sequence is found, one says that formula γ can be logically proved from premise α, or it can be deduced from α. In both cases, the formal notation is $\alpha \vdash \gamma$.

Among the conditions of proof-logical deduction, an axiom is such a theorem that is always believed to be valid in a particular model of the world, and this belief does not require any further proof. A tautology is a formula valid in *any* model of the world, regardless of its semantic meaning. For instance, schemas $\omega \vee \neg\omega$ or $\omega \implies (\omega \vee \beta)$ are typical tautologies; they are always true. While the conceptual axioms are *context-dependent*, tautologies are *context-independent*.

The only allowed inference rule in the conservative predicate calculus is modus ponens with implication (‘ $\implies$ ’) as the sole logical operation denoting necessity [Men79]. The application of this inference rule for a justification of a deductive step is verbally translated as: “*Formula* α_i *follows from the previous step* α_{i-1} *by virtue of modus ponens*”:

$$\begin{array}{ll} \alpha_{i-1} & \\ \alpha_{i-1} \implies \alpha_i & \text{premises} \\ \hline \alpha_i & \text{consequence} \end{array}$$

Chaining several inference steps of this kind can be expressed using the proof operator (‘$\vdash$’) to emphasise that, in general, a long chain of direct implications (‘ $\implies$ ’) may lie between the two formulas. However, the proof operator, in practice, carries the same meaning and can be replaced by the direct implication. This deductive law enables skipping the lengthy deductive chains and expressing the deduced consequence as a direct logical implication.

Such an abbreviation can significantly reduce effort in the future, when one starts deducing additional consequences. The newly formulated implication can be used similarly as other theorems of the logical theory. Thus, based on deduction it is possible to explicitly formulate new dependencies, to represent knowledge and rules. However, if such abbreviations are used as a justification of a reasoning step, it may confuse the external observer. This abbreviation may them be one of the reasons for labelling a decision as ‘spontaneous’ (as in [DD95]).

Logical deduction does not create new conceptual structures or generate *new* knowledge; it explicitly articulates *consequential* knowledge about the known conceptual objects. This analytical character of deduction can be used, for instance, in a qualitative modelling of structures [GI91]. However, some authors prefer to talk about causal links instead of deduction – the 'causal' terminology only emphasising the chains of direct logical implications.

In design the designer starts with a partial explicit requirement ($r_i \subset S$) of design problem $\mathscr{DP}$. Drawing on the domain knowledge, she believes that the design must contain a component represented as T. In other words she proclaims T a believed axiom of problem solving theory $\mathscr{T}^*$. Next, the logical dependency between axiom T and the explicitly required feature r_i must be proved; in other words, function r_i must be deducible from T and other believed axioms and theorems of theory $\mathscr{T}^*$.

If a logical proof $T \vdash r_i$ is found, then r_i is indeed a proof-logical consequence of the partial design solution T. By virtue of the completeness theorem [Lev74, Men79], solution T is also consistent with all deducible consequences (that is, $T \models r_i$). However, T may also imply other consequences (let us label them D) that were not among the initially explicated ones ($\neg(D \subset S)$). Since D is a logical deduction from axiom T, it is also consistent with it, but logical deduction cannot give any more information about the consistency between the explicitly desired feature r_i and the emergent ones (D). This consistency between the desired and the emergent is crucial for any further development of design, and must be formally assessed (see section 5.3.3). The 'desirable vs. emergent' functional dichotomy mirrors the case from sections 3.3 and 3.4 – with the desired goals, a solution often deliver additional, unexpected, and potentially undesirable consequences, which need to be filtered by articulating additional explicit constraints.

5.3.3. Admissibility – evaluation of logical consistency

In the two preceding sections, I showed how designers may tentatively articulate candidate solutions addressing a particular explicit problem specification using the abduction, and how they may analyse the consequences of the hypothesised solution by deduction. As section 5.3.2 revealed, some consequences are explicitly desirable, whereas

others are emergent. The latter have to be subjected to admissibility assessment, with respect to the explicit specification S.

The purpose of a logical evaluation of the solution and its consequences against the explicit requirements and constraints is necessary, in particular in design. First, because abduction is a speculative form of reasoning the abduced statements must be approved by a consistency check. Second, because the conflicts may be carried in during the construction of problem solving theory $\mathscr{T}^*$ by instantiating a generic domain theory DT for a particular conceptualisation $\mathscr{T}$ (see definitions in section 5.1). While generic domain theory DT may be consistent and sound, its interpretation in the particular conceptual terms $\mathscr{T}$ may reveal conflicts. An example of an occurrence of such 'instantiation conflicts' are physical contradictions in inventive problems [Alt84].

Furthermore, several component solutions $T_1, \ldots, T_m$ may be believed at the same time, each being perfectly sound. Each of them plays role in a particular module or sub-system addressing some requirements. However, when several independent modules are combined (that is $T = T_1 \cup \ldots \cup T_m$), clashes and conflicts among them may occur. Therefore, it is important to check the admissibility of the speculative abductions and partial deductions for consistency and mutual compatibility.

A technique suitable for the described purpose is known as *truth maintenance system* (TMS) [Doy79]. Basic TMS maintains the consistency through justifications; every new assertion of a theorem to a logical theory must be justified by a combination of already known (and valid) formulas. These formulae must be believed as axioms or they must be justifiable by other previously accepted theorems. In a recursive manner, the validity of the whole base of assertions depends upon an agent's belief of a small set of basic axioms. During the problem solving, certain formulas may be explicitly marked as 'false', not believed or proved invalid during tests. At that moment, a contradictory theory is avoided by the capability of TMS to trace the justification chains. The theorems are found that depend on, and follow from, the falsified formulas. These potentially contradictory and unsound statements that depend on the falsified ones, are 'removed', that is, also invalidated. In such a manner, any amendment of an agent's set of beliefs propagates through the problem solving theory, maintaining only the mutually consistent beliefs, until the soundness

and consistency of the whole theory is restored.

The basic idea was further developed into an *assumption-based truth maintenance system* (ATMS) [dK86b, dK86a] that is more powerful and removes many weak points of TMS. ATMS introduces so-called *contexts* bounded by the largest consistent sets of assumptions. *An assumption* is a novel concept here, and it is understood as a special kind of logical formula that expresses an agent's *tentative belief* of knowledge or fact (for example, a default value of some parameter). If a contradiction is encountered that may disturb the consistency of a logical theory with assumptions, the sets of assumptions, and consequently the contexts constructed upon such assumptions, are re-grouped. Context maintenance ensures that within any single context an agent's beliefs and the deducible consequences remain consistent. ATMS thus, handles the inconsistency by separating the conflicting assumptions into different contexts.

In design, suppose that one takes some elementary structural blocks as the 'believed assumptions' of the design theory, for example, 'mechanical spring' and 'hydraulic piston'. All deduced consequences in the design are justified by these assumed structural blocks and possibly other'proper' axioms, such as the theory of fluid motion. Assume that one of the deduced consequences from the problem solving theory with the above assumptions is found contradictory and undesirable (say, we require that the rigidity of suspension is adjustable, which is only possible in the hydraulic paradigm). ATMS thus splits the assumed structures into two distinctive contexts (that is, either mechanical, spring-based solutions or hydraulic, piston-based ones).

In other words, such a separation means that the basic building structures, as proposed initially, cannot co-exist, if the overall design solution is to be sound and admissible. ATMS can trace the hidden inconsistency back to the source through a perfectly straightforward, explicit operation, and divide the incompatible modules so that the soundness of a problem solving theory is restored. Additional examples of ATMS application in design are in [SCD+90, Tan97], and similarly motivated strategies also appear in [Alt84, SMW95].

5.3.4. Admissibility – compliance with constraints

The appreciation of the admissibility of a candidate solution also involves a check of its compliance with the known constraints. Ab-

duction and deduction discussed so far are concerned with the explicit requirements. Now, I devote some space to the other part of problem specification – *constraints*. Constraints are such conditions that must not be violated by the (partial) design solution under any circumstances [WAS95, MZ96]. Not violated condition is strictly distinguished from a satisfied condition: A condition denoted as λ is satisfied, only if it evaluates as true in a particular conceptual model. Condition λ, however, is not violated, if at least one of the following situations holds:

a. λ is computable in context $\mathscr{C}$, and it evaluates as true, or
b. λ is not computable at all in context $\mathscr{C}$.

Scenario (a) upholds the constraining condition and declares it "not violated" by virtue of a positive proof that it is explicitly satisfied. On the other hand, scenario (b) considers the constraining condition "not violated", because it cannot be proven explicitly that it is false. λ may not be computable, for instance, however, a kind of 'presumption of innocence' prevents the agent to declare its dissatisfaction or violation. Thus, it is possible to see the constraints placed on a particular artefact as borderlines that separate admissible solutions from the inadmissible ones. Such a border may be violated, if the candidate solution is clearly outside the designated space. Nonetheless, a border would not be violated if the constraint does not apply in a particular context.

There are numerous techniques for constraint satisfaction and validation [Kum92]. The computational power of constraints found its way into design, and there were several projects investigating various aspects of using constraints [BB97, Zdr97]. A typical area for constraint-based strategies comprises well-defined configuration problems, where a typical and benchmark representative of such well-defined tasks is an elevator configuration [Yos92]. Such a problem can be solved by deduction, truth maintenance and/or constraint satisfaction algorithms discussed in the previous sections.

5.4. Place of RFD model in design lifecycle

This section relates the RFD model presented in this chapter to the design lifecycle; to show, which phase of the design process my empirical work focuses on and what issues the RFD model addresses.

As I argued earlier, the majority of design approaches reviewed in chapters 3 and 4 address only the satisfaction of a given problem specification. My RFD model sees the explicit satisfaction as one of three intertwined actions expressed by predicates '*satisfies*', '*specifies*' and '*acceptable*'. Predicate '*satisfies*' was illustrated in section 5.3, and I will not investigate any of its operational or computational details in any greater depth in this book. The schema in Figure 5.2 represents one of several models of such an explicit design process found in the literature; this particular one takes into account different knowledge roles and knowledge sources according to the theory of generic task models (GTM) [TH93].

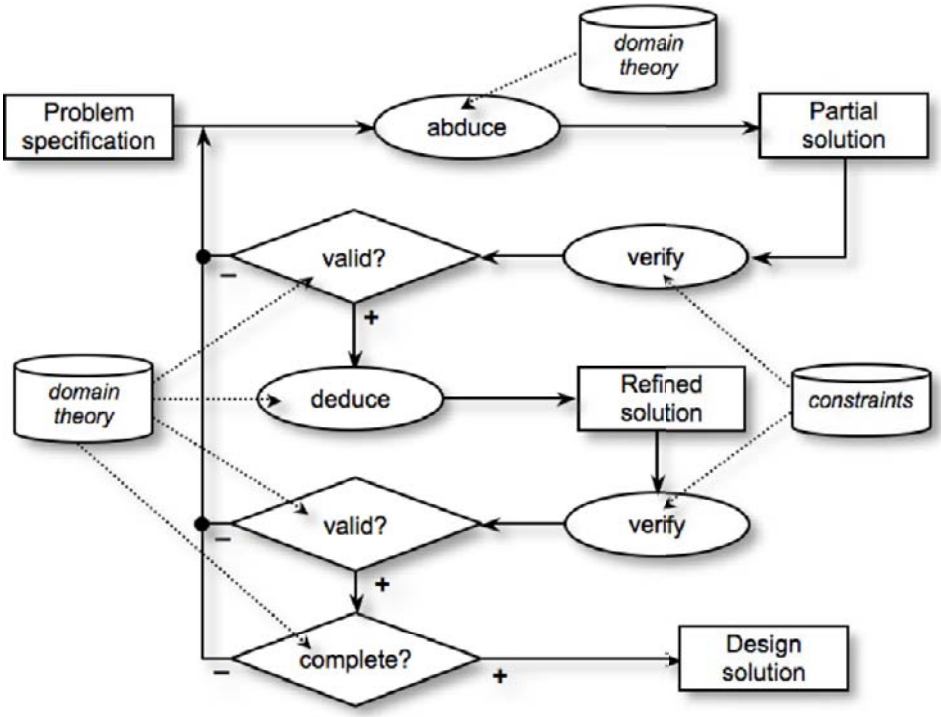


Figure 5.2. Generic task model of (explicit) design

Let me extend the GTM modelling predicate '*satisfies*' so that it also accounts for the remaining two predicates from my model. Several changes are needed to the GTM in Figure 5.2: First, we distinguish between a design problem ($\mathscr{DP}$) and its explicit specification (S). Consequently, we have to distinguish between a solution (T) to

such an explicit specification (that is, $T \models S$) and a solution to the actual design problem. My extension of the GTM contains one additional test for the solution acceptability: '$acceptable_{\mathscr{D}\mathscr{P}}(T)$'. Only those candidates that pass this check successfully, can be considered design solutions, that is, solutions to the original design problem $\mathscr{D}\mathscr{P}$.

If the acceptability check fails, it shows that the explicit problem specification ($S \subset \mathscr{S}^*$) or the conceptual foundation of the applied problem solving theory ($\mathscr{T}$ and $\mathscr{T}^*$, respectively) are not suitable. Consequently, the explicit problem specification may be amended to reflect the encountered gaps (that is, defining new sets $S' \subset \mathscr{S}^*_{\mathscr{N}}$), or the conceptual foundation of the problem solving theory may be extended (that is, set $\mathscr{T}^*$ may be defined). In some situations amendments of boths sets forming the design frame may take place. The extension of the GTM for design to include the theories introduced in this chapter is shown in Figure 5.3. The parts specific to my model, which are not present in other reviewed models, are drawn with thicker lines and in a different colour. The parts that are reused from the literature are now grouped into one operation that can be performed at the explicit level – the operation of satisfying the explicit problem specification (see label '*satisfies*' in Figure 5.3).

In my empirical validation I am particularly interested in those operations that interact with the 'cloud', with design problem $\mathscr{D}\mathscr{P}$. These are the operations of framing and re-framing that are at the core of this book. Taking into account the definition of a frame as a pair of two conceptual sets, $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$, the outcomes of the operation of re-framing are expressed by the thick arrows leaving the 'cloud' and labelled as $\mathscr{T}$ and $\mathscr{S}$, respectively.

According to the definition of the RFD model, the amended conceptual foundation $\mathscr{T}$ affects the articulation and deployment of a particular problem solving theory ($\mathscr{T}^*$), from which various candidate solution can be abduced or deduced. Similarly, an amendment to conceptual set $\mathscr{S}$ affects the explicit specification of the problem including the explicit requirements and constraints. Set $S \subset \mathscr{S}^*$ is used for the verification of the admissibility of any conclusion inferred from a particular problem solving theory. Finally, the dashed rectangle labelled '*satisfies*' may consist of different building blocks that depend on a specific approach (for example, logical design, design by analogy, or case-based design) – these are, however, not the topic of this book.

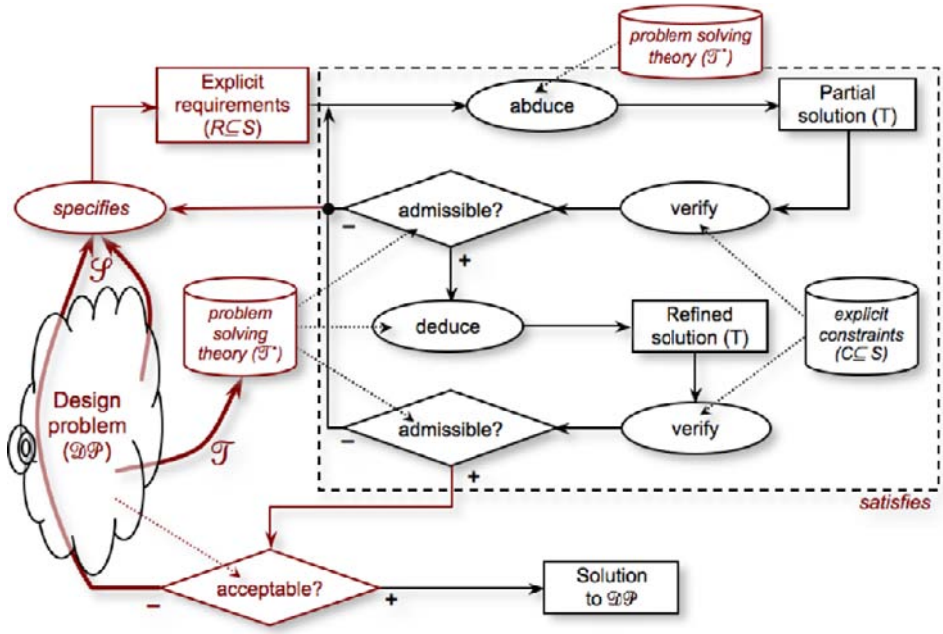


Figure 5.3. Place of RFD concepts in the explicit model of design

Let me now move to the experimental work in which the theoretical claims presented in this chapter are validated by actual design episodes. First, an overview of the experimental work is given in chapter 6. The annotation and analysis of the experimental sessions is given in chapter 7, and finally, further implications of the analysis are drawn in chapter 8.

Chapter 6

Background to Experiments

In chapter 5 a formalism was proposed to find out how reflective reasoning techniques fit the design process. The recursive model of framing in design using conceptual frames was presented as a natural extension of other design strategies working with explicit knowledge of design elements, algorithms and practices. However, the aspect of 'a natural extension' came out with a hindsight. It emerged from a number of experimental studies that were conducted to validate the theory of design frames [DZ01a, Dzb01, DZ01b]. The objective of this chapter is to acquaint the reader with the experimental background of the studies.

From the reviewed literature in chapters 3 and 4 we learned several lessons and identified some of the outstanding gaps. A particularly strong argument was made about the design not being a kind of intractable artistry [Dzb00b, Dzb00a]. This position was partially supported by experiments carried out by different researchers in the past [Can98, Cro97, KW96]. The 'artistry' was related to the designer's expertise, deep knowledge of the field, and the ability to transfer and adapt the knowledge gained in the past.

However, most of the studies in the literature either describe design projects requiring a number of collaborating designers or investigated unique, highly creative design tasks. In the first case, the focus was, understandably, on the issues of communication, co-ordination, and negotiation. None of these social aspects of design is the primary concern of this book, however. The studies from the second category are more informative, especially from the point of reflective reasoning. However, they focus on unique design situations that are

not so common in everyday practice. Such studies may unintentionally widen the common misconception that everyday design practice is relatively mundane and routine, whereas innovation and invention can occur only very rarely and in unrepeatable situations. And such a misconception could lead back to the point of a designers artistry, which I aimed to model.

6.1. Overview of the experimental study

To address the identified gap in the empirical research of design I conducted a study with the designers tackling complex, but not unique or exceptional design situations. The participants were design practitioners specialising in the design of controllers for large-scale systems. The problems they were asked to solve were formulated so that they could be managed within a maximum 12 hours long design session. At first glance, most design problems seemed to call for rather standard, if not routine solutions. However, the design problems allowed for some freedom; it depended entirely on the designers which path would be chosen. They were only expected to propose a solution they were willing to present to their peers and to senior colleagues. Formally, the assessment of the experiments was done in two steps:

1. validate the relevance of initial assumptions (generic assessment prior to any deep analysis), and
2. validate the principles of the RFD model ('model-specific' analysis of design sessions in terms of the RFD theory)

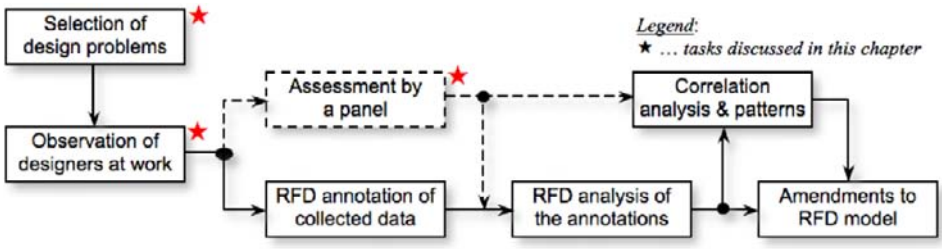


Figure 6.1. Overview of the experimental study and analysis

The sequence of the experimental sessions and their analyses is depicted in Figure 6.1. The first-pass evaluation of the observed and

recorded data is shown in the figure in the box labelled "assessment by a panel". The second step is addressed by the box labelled "RFD analysis of annotated data" and it runs independently of the assessment by the panel. The expectation is that the RFD analysis yields results that would correlate with the assessment by the panel, and allow formulating implications and amendments to the RFD theory.

6.2. Experimental sessions and methodological notes

To validate the theoretical ideas about modelling design frames, I facilitated a set of experimental design sessions. In the preparation for this experimental study, I was fortunate to receive support from one of the partners involved in a major European project coordinated by my research group. The representatives from this partner institution participated in the development of a new framework of reasoning in design, and one of their incentives was the fact that the research project in question followed a similar goal, with a greater emphasis on the collaborative and supportive tools.

Together we decided to conduct the experiment in the domain that was familiar and close to this partner's background, and chose designing controllers for large-scale systems as the primary domain. Altogether, 24 design sessions were recorded; each tackling a specific problem of designing a controller or control strategy. The assignments were not textbook examples but abbreviated extracts from real projects. My gratitude extends to the lab leader from the partner institution for sharing his extensive experience from the large-scale design projects. This was re-used in the form of design briefs for the experimental sessions.

Each of the experimental sessions was given a unique identifier (for example, "T11") that was used in the correlation analysis, summary, and conclusions (see section 6.3). In addition to a formal identifier, each session was named by the main artefact to design (for example, "shock absorber"). In tables 6.2 and 6.2 I give sample accounts of two problems that are annotated and analysed in-depth in chapter 7. In the descriptions of the sessions, I look at the following characteristics of the problems:

- expected primary goal(s);
- essential milestones of the designer's approach;
- quick description of the problem's character

Table 6.1. Description of the active shock absorber design session

T11: Active shock absorber (see also section 7.2)
Primary goals
◦ Design a shock-absorbing device that adapts to changing road conditions and driving styles
◦ The device reacts to disturbances caused by uneven road surfaces;
◦ Select applicable devices and respective controlled/controlling parameters (spring vs. pneumatic tubes, hybrid combination);
Essential milestones
◦ At the beginning, two different modes of operations considered – manual and automated adjustment of the shock dampening characteristics;
◦ Problem attended to in two parallel contextual worlds, the difference between them growing as the solution is refined;
◦ Selection of a pneumatic approach where dampening characteristic depends on pressure rather than material of the absorber;
◦ A set of assumptions formulated to clarify the requirement of 'active adjustment of chassis clearance' from the initial problem specification ('active' in interpreted differently for shock absorption context and for chassis context);
◦ Eventually, a compromise is considered and different contexts are merged
Problem character
◦ Task is clearly solved in several separate contexts (manual vs. automated tuning, shock absorption vs. clearance);
◦ The separate contexts are characterised by different physical principles and components suggested in each particular case;
◦ Presence of multiple contexts resolved by separate controllers for specific situations with a hierarchically superior switchboard activating sub-modules

A typical assignment of the set design problems included the design of a control strategy, a control loop with the controller, or both. All set problems involved technical or technological systems. That is, I did not test the RFD theory on the popular tasks, such as house interior design or architecture. Architectural design was a favourite research topic in the past decade, and many high-quality studies were done in this area [Fis92, Ger90, GdSGM96, Sch83, WP97]. Neither did I include sociological or economic design problems, such as described in [Sch83]. My primary attention was on technological systems, devices and controllers, where the referential literature includes [Alt84, BG94, BC85, CMI+98, CB99, IFVC93, Sch83, SCD+90].

From the methodological point, the studied design sessions were focusing on early, conceptual phase of design. The most serious implication of such a focus for the designers was restricting them to the use of sketchy drawings on paper, conceptual descriptions and justifications of their decisions in words. They did not have any

powerful modelling packages at hand, such as Matlab/Simulink™ and CAD tools. However, unlike similar conceptual design studies [CB99, Cro97, Cro99], our participants were given a computational tool to structure and record their reasoning on-the-fly. The presence of such a tool can be a double-edged sword; it may have positive as well as negative effects on the designer's performance [Dav95]. Since the experiment setting is crucial for the result validity, let me explain the usage of the customised computation tools in the experiments.

Table 6.2. Description of the paper smoothing plant design session

T21: Paper smoothing plant (see also section 7.4)
Primary goals
○ Design a plant for achieving a vaguely defined functionality (smoothing raw paper);
○ Refine the vague specification with various quality- and process-related requirements (thickness reduction, prevention of paper damage, simple maintainability, etc.);
○ Select a suitable physical principle and subsequently layout of the plant;
Essential milestones
○ Early solution with a pair of rolling drums applying pressure on paper thus reducing its thickness and smoothing its uneven surface;
○ To comply with quality constraints, pre- and post-processing units added (dampen paper before rolling and dry it afterwards – lower risk of tearing);
○ Articulation of restrictive assumptions and additional constraints (for example, regarding plant dimensions);
○ Re-formulation of the linear layout to a zigzag layout of the rolling drums;
○ Principle of rolling considered in a broader context (re-interpretation of rolling from pressure application to abrasion; instead of drum pairs a zigzag layout of single rolls found sufficient);
○ Incorporation of pre- and post-processing units into feeding and driving units
Problem character
○ Task starts relatively routinely, then involves an innovative sequence of reformulations);
○ Simultaneous design of plant layout, structure and control strategy;
○ Oscillation between refinement of a vague specification and solution development;
○ Initial solution getting more complex as additional constraints and assumptions considered; after re-interpretation the complexity radically reduced

The advantage of the ethnographic methods is that the experiment is conducted in the setting that is natural for the participant [JRC+00, RSP95]. In other words, the investigator takes part in the

everyday activities of the participants, and typically acts as an observer or interviewer. Different ethnographic metods include video and tape recording, note taking, looking over the participant's shoulder, or thinking aloud [AC93, KMM93], and all can be applied 'on-line' or 'off-line'. In the former case, the experiment is overt and the observers have an active role. They may ask questions, as soon as an unclear decision or an unexpected line of thought is observed. Alternatively, the observer works covertly, without disturbing the participants. In this case, the ambiguities are clarified at the end of the experiment, when the investigator goes through the notes, and interprets the observations in an interview with the participant.

It is commonly accepted that the on-line methods are better, when it comes to the precision of the observation [ACM93]. However, because the observer directly interacts with the designer, and intervenes in the design, the validity of the observation may suffer [Dav95]. The interventions of the observer may alter the natural setting of the experiment, and the recorded process may differ from the situation, if the designer was undisturbed. On the other hand, the off-line approach preserves the natural setting, and minimises the observer's interventions. However, because the ambiguities are discussed and justified after the session, the participant may give a different line of reasoning to justify decisions compared to the one that actually occurred during the design.

I decided to take a combined approach to capturing participants' reasoning and justifications in design. In practice it meant that the participants worked on their own, without any direct intervention by the investigator. They chose their own pace, strategy, and levels of granularity. Simultaneously, they were asked to record their progress using a networked computational toolkit that enabled external observers to follow the explicit portion of their design reasoning. Such a set-up also gave the designers an opportunity to post a query or an interesting idea to the debating environment. From there, any statement could be picked up by the observer, who could answer it or provide a comment on it.

This form of interaction was chosen after the initial interviews with the domain expert from the participating organisation. The participating designers were used to interacting with their peers on an informal level, but preferred a more formal intervention from the senior staff, tutors and customers. When we later interviewed the par-

ticipants and raised this point, they explained that formal interaction gave them the opportunity to acquire the requested information and to have a record for future reference.

The participants recorded their notes in an on-line notebook. Each statement they entered was organised into *threads* [SBS98], if it extended a previously recorded point. A thread contained the reasoning step behind one design decision or a set of related decisions, where the decision was concerned with a specification of the design task or with a generation of the solution. New threads were started, when the designer's focus moved to a different module, beyond the scope of the explicit context defined at the particular moment.

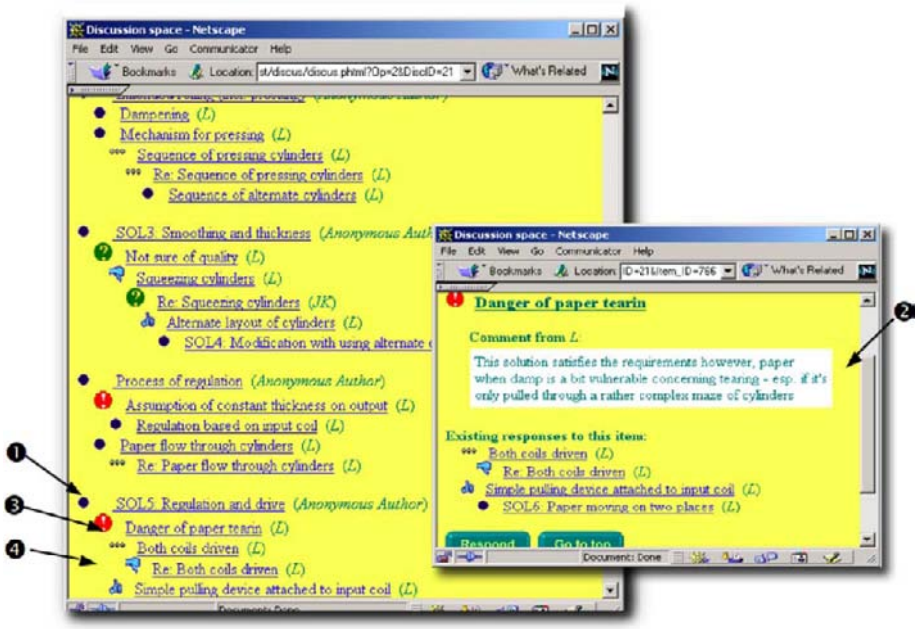


Figure 6.2. Threaded justification of reasoning steps in a design session

A snapshot of the environment recording the justified reasoning is shown in Figure 6.2. The right-hand window shows the details of one such justification, and the left-hand one contains the sequences of reasoning steps. Threading is a useful vehicle for tracing the development and extension of a particular reasoning chain. Figure 6.2 shows the situation when the designer discovered an unexpected flaw in his approach to smoothing paper (details in section 7.4). The danger of

damaging the paper was noted in the active thread (see label '1'), and detailed in the window labelled as '2'. A follow-up trying to fix the conflict appeared that was unacceptable (label '3' and 'thumb-down' icon). Then, another, acceptable modification was proposed (see '4' and 'thumb-up' icon).

Another tool available to the designers aimed to capture the co-development of the problem specifications and solutions. In line with chapter 5, the tool in Figure 6.3 defined 'the context' (see label '1') as a selection of explicit requirements and constraints (in the left pane, labelled as '2') articulated by the designer in respect to the designed artefact. The explicit commitments to a certain problem specification were accompanied by articulations of respective solutions addressing the particular design context (pane labelled '3'). 'Active context' was incremented automatically whenever the designer proposed a new (partial) solution. Explicit details of the commitment to a particular requirement or solution are displayed in the bottom right-hand pane and label '4' points to the details of one such requirement 6.1.

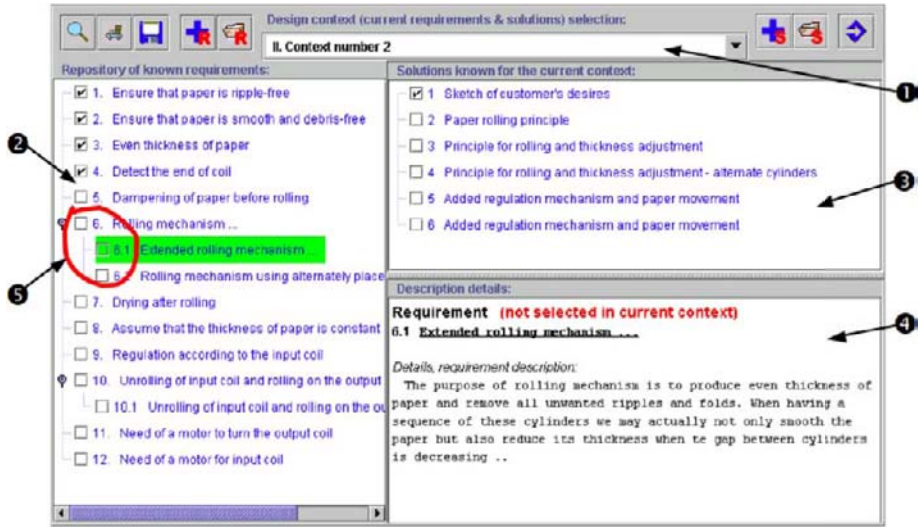


Figure 6.3. Design context capture during the design session

The snapshot in Figure 6.3 shows design problem T21, which is annotated and analysed in section 7.4, made when the designer returned to a previously attended context II, after articulating additional extensions. The designer kept several lines of enquiry open at

the same time, but grouped them into separate contexts. As the design progressed, the lists of explicit requirements and solutions grew: New requirements could extend the old ones (requirement 6.1 refines requirement 6; see pointer '5'), or address a new feature (for example, requirements 7 or 8). A detailed transcript of records of such a co-evolution for design session T21 is annotated in section 7.4, and further details of the capturing tools are in [DZ01a, Dzb01].

The main utility of these two tools was that they collected data about design reasoning in a structured form, and simultaneously provided the observer with a discrete opportunity for intervention. The intervention could be masked as a design comment or recommendation within a threaded sequence. It fitted well the designers' work style, and did not distract them by demanding them to stop their reasoning and answer a query from the observer. This 'natural design evolution' would be impossible, if the observer's interventions were formulated verbally and presented directly in person.

Taking into account the non-invasive data collection, and the automation of threading and context creation I believe the danger of imposing the investigator's opinion onto designers was minimised. The recordings of the reasoning steps became the primary data source for the analysis. However, to elicit participants' subjective opinions post-session interviews were conducted too. During these interviews each session was summarised to highlight the main milestones in its interpretation. The post-processing phase was helpful because of the use of multiple media for capturing the designer's reasoning; including paper notebook and pencil. Debriefing merged data from the separate sources into a single design case, and included the location of exact places where the designer moved from words (computational tools) to sketches on paper, and vice-versa. The information about these shifts in reasoning was, in turn, useful for the validation of my RFD model of the design process.

6.3. Assessment of the experiment prior to the analysis

Assessment of the recorded design sessions by a panel of experts was conducted to validate the selection of the design tasks for the experiment. It also validated the correctness of my initial assumptions of what features characterise design as a problem solving activity. Thus, the panel was asked to assess the collected data records (prior to any

analysis) focusing on the scope and criteria from section 6.3.1 below. Section 6.3.2 provides a brief summary of these opinions in a tabular form, and in chapter 8, these subjective opinions are compared with the results of the RFD analysis, which is described in chapter 7.

6.3.1. Scopes and criteria of assessment by a panel

The following main categories (explained in more depth below) were used to characterise the problems:

1. Presence of the phenomenon of co-evolving design specifications and solutions;
2. Reflection on partial design solutions;
3. Re-formulation of design problems and articulation of additional requirements and constraints;
4. Presence of knowledge transfer by re-using familiar frames

The list of categories below serves to refresh the memory of section 3.4, and summarises what was looked at under each category. The subsequent paragraphs associate the above-mentioned categories with the specific design features to provide context for their analysis:

Co-evolving specifications and solutions

This category considers the hypothesis that problem specification (both requirements and constraints) evolves in parallel with the design solution. This means that design is iterative and sequences of reasoning steps and decision amendments should be observable. Second, the co-evolution means that the iteration is not necessarily driven by a completeness of the problem specification. Iterative steps may be opportunistic and exploratory, and may be conducted to acquire deeper understanding of design behaviours or unexpected results.

In other words, we expect a richer interaction between the problem specifications and solutions than can be explained from simple operational models of design. In many models, the amendment of a problem specification is only allowed if there are explicit violations by a proposed solution. In my opinion, in addition to the explicit evaluation there are also internal evaluations, ‘exploratory probes’, and we expect to observe such probing through confirmation and rejections of specifications and solutions.

Reflection embedded in design

This category follows the point that reflection is an attempt to uncover and rectify the unexpected behaviours through exploratory probing. What we want to observe is that the internal, tacit evaluation occurs continuously as the development of the design solution and specification progresses. This form of assessment would address the partial solutions and assess them not for their logical soundness but acceptability and desirability. Thus, design should exhibit 'breakpoints' where solution acceptability is checked in addition to the explicit truth and consistency maintenance. Metaphorically, this situation is similar to the practice of software developers: Any code, which passes through a compiler, is sound and may serve as a developmental build, but not all builds enter the evaluation phase. Only those that are acceptable to the developer may act as 'released versions', which are then presented to the external world.

In this category we aim to observe such tacit checks of solution *acceptability* contrasted to the solution *admissibility*. Reflection may occur with incomplete artefacts or solutions, for instance, to test a particular probe, assumption or approach. In such scenarios, reflection would aim to validate and refine the existing problem specification that may be vague, incomplete or ill-defined. Thus, reflection should typically lead to a re-formulation of the design problem.

Re-formulation of the design problems

This objective addresses the assumption that problem specification in design is subject to development, refinement and extension. Design problem taken from different angles should lead to the presence of certain dynamics in the designer's understanding and interpretation of it. Such dynamics should be manifested through discoveries and articulations of new conceptual terms specifying the design problem and the concepts for the construction of solutions. I do not expect that these two conceptual sets would be fully defined and enumerated during the design process.

On the contrary, the designer is expected to attend to the selected sub-set of the problem specification and propose partial and conceptual solutions verifying the relevance and completeness of that particular focus. It can be expected that any design context yields both expected and unexpected results. Subsequently, the unexpected

features will need to be explained and rectified, whereby new requirements, assumptions, constraint or structural concepts should be brought into the design to enable such explanation. In other words, the design problem may need to be re-formulated or re-interpreted, and evidence for such actions should emerge in the captured sessions.

Knowledge transfer and familiarity

This objective aims to observe the application and re-use of familiar vocabularies when framing the design problem. Familiar and past design cases, experience and analogy-based reasoning are expected to be important knowledge sources and reasoning strategies, and they should be observed in the interpretation and (re-)formulation of the design problems. A knowledge transfer may occur between the similar cases both tackling the same issues (so-called 'in-domain transfer'), or it may be inspired by an interesting case that is similar on an abstract conceptual level (that is, 'cross-domain transfer'). Some evidence is expected of the designers transferring chunks of the previous experience to refine or extend both design specifications and solutions.

In general, we seek a flexible application and re-use of different knowledge sources and the capability to familiarise oneself with the current problem using the past experience. This expectation should trigger the presence of such operations as retrieval of familiar cases or adaptation of a re-usable component. Various design prototypes, device patterns or models, and the articulation of additional conceptual primitives by analogy should dominate over rules and algorithmic procedures. We also expect that such prototypes and familiar scenarios would be useful for the (re-)interpretation of the design problem.

Assessment criteria

The following criteria were agreed with the panel to characterise each experimental session and to facilitate correlation analysis. I list them as pairs of ⟨*evaluated feature*⟩ and a {list of possible values}:

⟨ ***Overall impression*** ⟩ ∈ {routine, partly innovative, innovative};
⟨ ***Form of problem specification*** ⟩ ∈ {detailed, complete without details, functional outline, vague functionality};
⟨ ***Dominant source of solution*** ⟩ ∈ {prototype-based, in-domain transfer, cross-domain transfer};

⟨ ***Conceptual abstractness*** ⟩ ∈ {high-level abstraction, conceptual abstraction, implementation};
⟨ ***Amendments in problem specification*** ⟩ ∈ {modifed frame, problem re-formulation, extension, clarification};
⟨ ***Prevailing operational method*** ⟩ ∈ {analysis, synthesis, balanced analysis and synthesis}

The criterion 'overall impression' takes a broad picture of the particular design task. It covers aspects like the nature of the designed artefact, re-use of previous knowledge, novelty of the solution, presence or absence of expected and unusual decisions during the design. A task was 'routine', when no unusual decisions were observed or the design followed a standard algorithm. 'Partial innovation' meant that in some aspects, the design exhibited an uncommon feature or knowledge transfer. 'Innovative' problems contained an off-the-beaten-track approach and corresponded to major improvements of the existing control systems.

Under the criterion 'form of problem specification', the completeness of the design brief was assessed. The four values reflect the increasing vagueness, starting with a complete specification with all the facts needed to proceed with the design. Next, it was a specification with basic conceptual requirements but no details such as dimensional data or chemical compositions. A more abstract specification was labelled as a 'functional outline', describing what was expected from the designed artefact in functional terms. The problem specification marked as 'vague' permitted alternative interpretations, and only hints about the expected functionality were given.

In the 'dominant source of solution' criterion, the prevailing knowledge source used to develop the solutions was assessed. That is, the different types of knowledge transfer and re-use occurring in design were looked at. The first group contained the cases where clear prototypes existed, and these did not require any amendments. The second group was marked as 'in-domain transfer', and contained the cases where an analogous reuse at a medium conceptual level was observed. Previous cases were taken from the same or conceptually similar problem domain (say, heat propagation was analogous to air conditioning). Finally, 'cross-domain transfer' occurred at a high conceptual level; the analogy was inspirational, and not common (say, measurement of electric resistance using the ball pressure instead of the conductivity).

The 'conceptual abstractness' looks at the focus of the design. The solution might have been developed at a highly abstract level, with little attention to the structural details of the high-level modules. Alternatively, the designer developed medium-level conceptual terms and attended to some implementation issues. The details of the realisation were limited to qualitative concepts, such as safety, user comfort, or user friendliness. Finally, the most detailed solutions went to the level of system implementation, attending in detail to qualitative and quantitative parameters.

The experimental sessions were also assessed on the number of changes required to formulate a specification reflecting the actual product being designed. At one extreme, a significant re-formulation took place, and the problem needed to be completely re-interpreted. On the medium level, limited additions or minor amendments were observed that did not change the principal points in the overall frame. Finally, the opposite extreme was marked as 'clarification', and included those tasks, where no re-formulation was necessary. Typically, the amendments were restricted to the translation from the customer's language.

The last criterion attended to the dominant operational method. The panel was asked to comment, whether the particular problem forced the designers to approach the problem by analysing it, or whether they applied a constructional, synthetical attitude. An alternative between these was a 'balanced interplay' of analysis and synthesis. For instance, new analytical knowledge was obtained through the synthesis of a sub-component, or a specific component was added based on the analysed behaviour of the incomplete artefact.

6.3.2. Summary of the assessment by a panel

As mentioned in section 6.3.1, each experimental session was assessed by six criteria. The sessions were judged only from the captured data, the electronic design protocols and paper-based sketches. The panel consisted of design students, practitioners and theorists, and covered a range of expertise in engineering design and in the design of controllers. Two evaluators were university graduates with a major in control systems, one was a design practitioner with several years of experience. Also invited were three professionals with a general overview of the design research. And to complement this group, a cognitive scientist was invited to balance the expertise of the panel.

The assessments were compiled into tables that served as a basis for the correlation analysis. When compiling the judgements, I counted the individual responses and the panel response was determined by the most frequent judgement. When two values had the same occurrence, both were accepted as valid scores. Such ambiguities were expected from the ill-structured problems we were investigating. For instance, design task T11 (shock absorber, section 7.2) was judged on the criterion 'form of problem specification' as having 'incomplete functional outline'. However, it had two values, 'in-domain transfer' and 'cross-domain transfer', when the source of solutions was looked at (see Table 6.3).

A dual valuation typically occurred when two or more distinctive stages of design were observed, and all of them were equally important. For instance, an interesting reasoning step could appear through an inspirational, high-level transfer of past knowledge. This particular shift of a perspective was instrumental in formulating the final solution, however, the task as a whole was judged differently. A typical example is task T3 (ball and horizontal rail) that exhibited a typical prototypic application of resistance measurement, but at the end, a more innovative approach superseded the prototype.

Table 6.3 shows a correlation between the selected criteria that were used for evaluation by the panel. It shows the relationship between the different criteria for the ill-structured design problems. The highlighted cells in the table represent a re-occurring pattern in the assessment. Where two cells of the same attribute are highlighted, it means that value A, value B or both were observed. For example, if values 'functional outline' and 'vague outline' are highlighted, it means that either of them accompanied innovative tasks. Table 6.3 shows coincidences of the following assessments that emerged from the panel's collective opinion using *shaded* background:

- Overall design process: potential for 'partial' or 'significant' innovation;
- Problem specification: 'functional' or 'vague' outline;
- Solution source: 'in-' or 'cross-domain' knowledge transfer;
- Conceptual details: 'high-level' or 'moderately abstract' solutions;
- Changes to problem specification: 'significant amendments' or 'extensions';
- Operational method: 'analysis and synthesis in an interplay'

A complementary selection described by the statements below is shown in the same table using *striped* background. This coincidence highlights the complementary values compared to the *shaded* cells. However, strictly speaking, the coincidences are not exclusively disjunctive. There are cases conforming to both patterns, for instance, case T3 that exhibited both routine and innovative patterns. The overlapping assessments are depicted with a **diagonally striped** background. This complementary correlations (with *striped* background) take into account the following assessments:

- Overall design process: potential for 'routine' or 'little innovation' outcomes;
- Problem specification: 'complete' or 'some details missing';
- Solution source: 'prototype re-use';
- Conceptual details: 'moderately abstract' or 'implementational' solutions;
- Changes to problem specification: 'minor amendments' (clarifications);
- Operational method: more 'synthesis' than 'analysis'

Taking into account relative occurrences of the individual valuations, in some cases, there was a clear 'winner'; in others, the differences were less striking. For instance, a majority of tasks was routine (37%) or somewhat innovative (41%). A similar distribution over a much larger base of cases is reported in [Alt84], where only a fraction of all designs is groundbreaking; most are minor or medium improvements. In my experiments, the initial problem specification usually fell between the 'functional outline' (38%) and 'not detailed but otherwise complete' category (35%). This shows that it is rare to have a completely specified problem, and also that too abstract specifications do not facilitate design. A 'moderate' level of specification details seems to be more suitable for innovation than deliberate woolliness: Too detailed specification restricts the freedom of exploration; the vagueness may complicate the formulation of the first conceptual frame.

The majority of cases exhibited a direct transfer (39%) or an adaptation of an existing prototype (43%). A clear winner in the conceptual details of solutions was the 'medium level' (64%). These two findings support the observed rarity of designs that alter the conceptual foundations of a specific field, as argued in chapter 3. Most tasks

upheld my assumption of a co-evolving problem specification and solution; typically, new design requirements were added to or extended the initial specification (44%). Most problems were tackled by an interplay of analysis and synthesis (46%), which supports the arguments made in [LS92, Oxm90, SCD+90]. As mentioned in section 3.1, trying to split a task into analysis and synthesis is often trying to differentiate the indistinguishable. Even in the prototypic cases, the interplay was clearly observed.

Table 6.4 shows a summary of the opinions of the panel in respect to the quality and complexity of the designed product. The first category assesses the designed product, and the values range from the design implementing a routine controller and focusing on the control algorithm, through customised controllers, and ending with designs of both controlled systems and controllers. In the category of design completeness, the allowed values were: a single detailed solution, a single conceptual solution, several alternative solutions, and abstract conceptual ideas. This second table is used later in chapter 8 to show interesting patterns in the analysed results. The *shaded* cells in Table 6.4 cover those tasks that were judged as generally 'algorithmic' and yielded rather simple solutions.

I should stress that these relative co-occurrences were obtained prior to the analysis of the captured data in terms of my RFD theory. Therefore, I return to these correlations later, after analysing the collected data by the RFD theory. To support my theory I expect that the tasks in *shaded* cells in Table 6.3 would exhibit more radical shifts and amendments in the respective design frames. A similar finding will be presented for the categories in Table 6.4 (that is, cells with *shaded* vs. *white* backgrounds).

The details of the RFD analysis are in chapter 7, where I provide examples of how the collected data was interpreted in English and translated into the language of RFD theory, to see if the RFD model is a realistic explanation of the collected and annotated data.

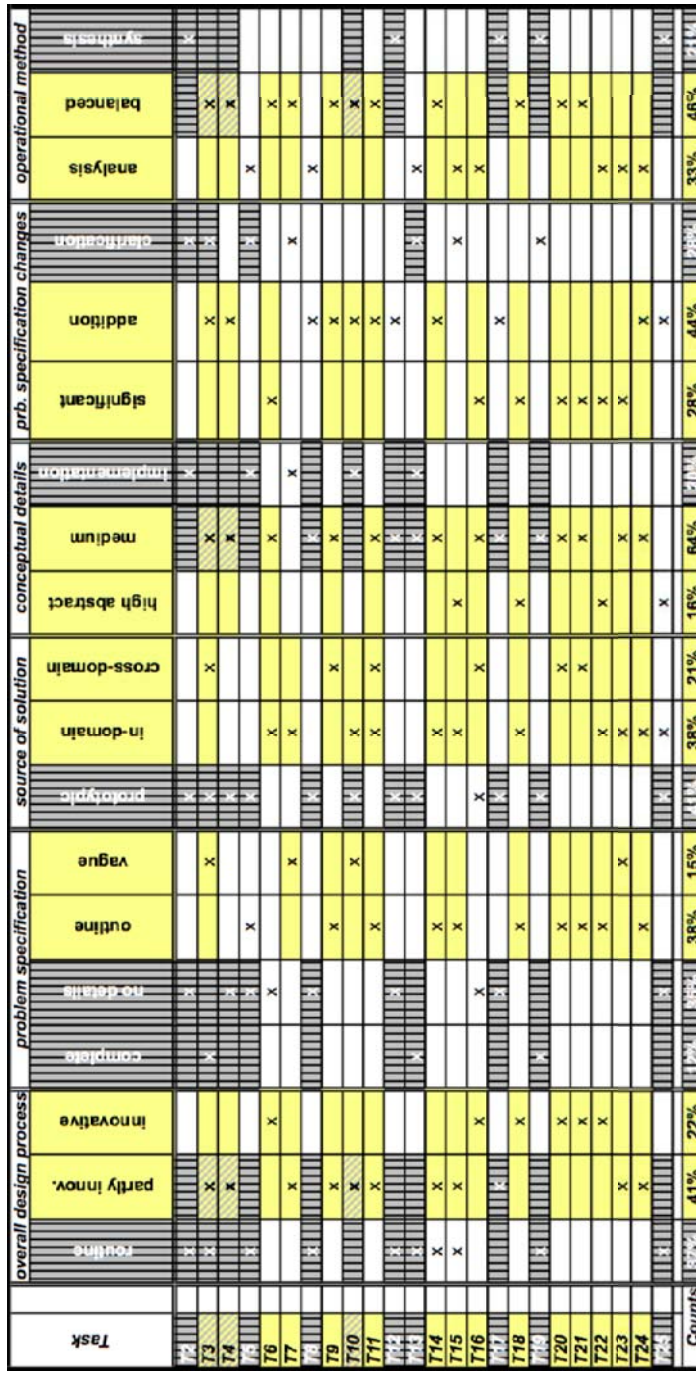
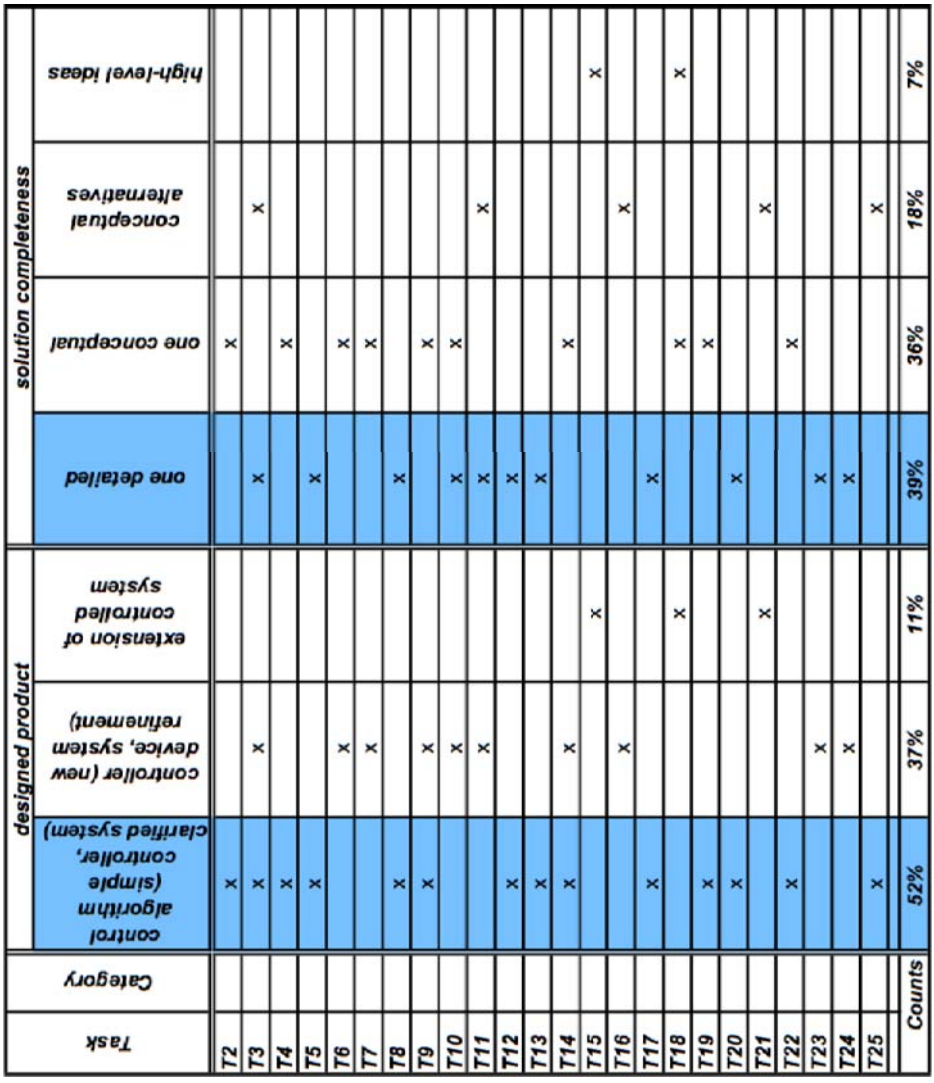


Table 6.3. Correlations in panel assessment: innovative approach = ill-specified problem + solution transfer + co-evolution of specification and solution



Task	Category	designed product				solution completeness		
		control algorithm (simple controller, clarified system)	controller (new device, system refinement)	extension of controlled system	one detailed	one conceptual	conceptual alternatives	high-level ideas
T2		x				x		
T3		x	x		x		x	
T4		x				x		
T5		x			x			
T6			x			x		
T7			x			x		
T8		x			x			
T9		x	x			x		
T10			x		x	x		
T11			x		x		x	
T12		x			x			
T13		x			x			
T14		x	x			x		
T15				x				x
T16			x				x	
T17		x			x			
T18				x		x		x
T19		x				x		
T20		x			x			
T21				x			x	
T22		x				x		
T23			x		x			
T24			x		x			
T25		x					x	
Counts		52%	37%	11%	39%	36%	18%	7%

Table 6.4. Panel assessment of the complexity, completeness and quality of design solutions

This page intentionally left blank

Chapter 7

Annotation and Analysis of Experiments in Designing

The purpose of this chapter is to acquaint the reader with the experimental work to validate my Recursive Model of Framing in Design (RFD theory). In sections 7.2 and 7.4, it is shown how the raw data was interpreted as design episodes interspersed with the explicit proposals of partial solutions to the explicit problem specifications.

After annotating the collected data the two tasks are analysed in sections 7.3 and 7.5. The analysis is conducted in terms of the RFD theory presented in chapter 5. Each of the two analytical sections is concluded with a summary of reasoning episodes that were identified in the interpreted data. These summaries are then used in chapter 9 for a modification of the RFD theory. Figure 7.1 shows which phases of the experimental study are tackled in this chapter.

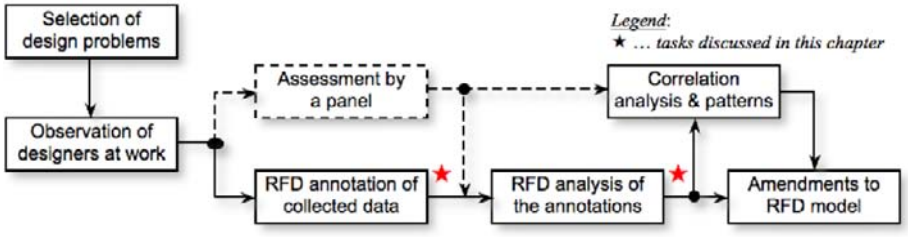


Figure 7.1. Phases of the experimental study tackled in this chapter

7.1. Principles for the annotation

Before analysing the annotated experimental scenarios, let me define the principles for the annotation. These principles reflect the knowledge-level actions introduced in chapter 5, and their purpose is to provide a succinct account of each analysed session in the form of a table that relates the individual design episodes, the frequency and the order of the investigated actions. The same principles are used in chapter 8 to discuss the aggregated results from all the experiments, and the patterns that occurred in the experimental sessions.

In the annotations, terms *design task* and *design session* both refer to one particular design problem tackled by the designer over a certain period of time. Each design session started with the presentation of a *design brief* to the designer. The brief was written in a customer's language, and its purpose was to express the requirements and expectations from the designed product. These briefs were consulted with the panel of experts (also mentioned in chapter 6).

Each design session lasted for several hours and contained several explicit milestones. The designer articulated a milestone to record his *explicit commitment* to a particular problem specification or to a partial solution. One outcome of each explicit commitment was a definition or a re-definition of the *design context*. A design context consisted of a list of selected (attended) *requirements*, *constraints*, *assumptions*, and other statements that clarified the designer's position in respect to the problem interpretation. In addition to the explicit specification, a context may have contained a *candidate solution*. The evolution of explicit commitments and design contexts was captured in full, but is only presented here in an abbreviated form.

However, records of the evolving design contexts are too coarse-grained, and only show the design problem at the moments when the designer made an explicit commitment. Design contexts do not record any reasons, justifications or alternative choices considered by the designer. Hence, to complement design contexts, I use term '*design episodes*'. The episodes consist of *reasoning steps*, and they act as the transitions between the explicit commitments from different design contexts. Typically, there were two design episodes between two consecutive design contexts in the early stages of a design session. Toward the end of the session, there was usually one episode connecting two explicit commitments (usually solutions).

Each design episode describes a chunk of the designer's reasoning, and is associated with a thread from the justification records. I took into account not only the syntax of the recorded threads but also their content and focus. Thus, a single design episode could be concerned with the elaboration of a particular viewpoint, analysis and comparison of the alternative viewpoints, product decomposition and development of the sub-components. The identification of boundaries between the design episodes was informed by the order in which the justifications were recorded.

Finally, the reasoning steps of a design episode correspond to the sequence of the knowledge-level actions summarised in Table 7.1. These actions follow from the sequence of modelling predicates of the RFD theory from chapter 5.

Table 7.1. Definition of reasoning steps forming a design episode

Label	**Verbal description of referenced action**
1a	Initialize design frame Φ (sets $\mathscr{S}$ and $S \subset \mathscr{S}$, such that $specifies(S, \mathscr{D}\mathscr{P})$)
1b	Articulate design frame (sets $\mathscr{T}$ and $\mathscr{T}^*$)
2	Retrieve similar problem frame ($\Phi_{SIM} = \langle \tau, \theta \rangle$)
3a	Knowledge re-use (articulate $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$, where $\mathscr{S}$ similar to θ)
3b	Knowledge re-use (articulate $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$, where $\mathscr{T}$ similar to τ)
4	Are re-used, adapted sets T, $\mathscr{T}$ and S, $\mathscr{S}$ admissible?
5	Find candidate solution ($T \subset \mathscr{T}^*$: $satisfies(T, S)$)
6a	Develop S, $\mathscr{S}$ into details (incl. alternatives, refinements)
6b	Develop T, $\mathscr{T}$ into details (incl. alternatives, refinements)
7	Is candidate solution T acceptable?
8	Re-interpret conceptual frame (i.e., $\Phi \rightarrow \Phi'$)

Reasoning steps *1a* and *1b* are assigned to the episode that features a formulation or a selection of design objectives and objects, respectively. These two steps represent an explicit commitment to the requirements, constraints, and conceptual primitives. Usually, they are associated with an articulation of a new design context. Reasoning step *2*, and consequently *3a* and *3b*, represent any reference the designer makes to the previous design cases, domain theories or design prototypes. These steps typically occur together as knowledge retrieval and knowledge re-use. 'Keywords' that can be found in the justifications include, for instance, "*from the theory*, we know...", "...to create a *typical* control loop" and similarly.

Reasoning step *4* annotates the explicit assessment of admissibility. An admissibility check is assigned to the episodes, in which clear criteria existed against which the designer evaluated his partial solutions, compared alternative approaches, or checked the compatibility of the re-used design knowledge. Step *5* corresponds to the articulation of an explicit solution, solution model or a partial solution. Typically, this step was accompanied by a sketch containing the components, behaviours or functionality of the solution. Similarly to steps *1a/1b* this step was captured in the records of explicit commitments and design contexts. Reasoning steps *6a* and *6b* correspond to the elaboration of a particular design context. They include the extension of the partial solutions, the clarification of design objectives, and various deductive refinements in the problem specification or solution descriptions. They differ from steps *3a/3b* in the fact that no specific reference is made to past knowledge, nor do any new conceptual terms appear in these refinements. Unlike steps *1a/1b* that merely state the facts explicitly, steps *6a/6b* tend to show the refinements in the form of causal or deductive chains.

Reasoning step *7* is assigned to those episodes that exhibit the signs of evaluation, although there were no specific and clear criteria defined at the beginning of the episode. Typically, new parameters or objectives emerge from the episode featuring this acceptability check. Finally, step *8* is a reference to the design perspective shift. In other words, when *new concepts* emerge in the design episode as a result of reflecting on the proposed designs, the episode is considered as a problem re-framing. Often, frame amendment is accompanied or followed by an introduction of new assumptions or preferences and a tentative commitment to them. Thus, we may see step *8* as the creation and application of an exploratory probe.

Let me now annotate design session solving the problems of active car suspension (section 7.2) and of paper smoothing (section 7.4) by splitting the design sessions into respective design contexts and design episodes. Then, sections 7.3 and 7.5, respectively, interpret the annotated episodes using the RFD.

7.2. Annotated task T11 – active suspension

The initial design brief given to the designer for this problem is as follows: *Suspension and shock absorption play an important role with*

regard to the car passengers' comfort. Design an active shock absorber and suggest a control strategy, so that the suspension mechanism would be able to adjust according to the current state of the road. In addition to the absorption of shocks caused by an uneven road, we would like to adjust the chassis clearance, so that the car has the optimal aerodynamics for a given type of surface.

The transcribed design episodes contain abbreviated references to three elements that are detailed in the appendices; namely: *Req-X* stands for requirement number X, *Sol-Y* stands for solution number Y (both detailed in Appendix A, and *J-XYZ* points to the justification record number XYZ in Appendix B.

Design episode T11.a (design context DC-I)

Initialisation of frame DC-I using similar conceptual frames: The designer began with an initial task analysis that was based on his knowledge of existing approaches to car suspension. First, he expressed term 'active suspension' using three tenets in a language similar to that of the initial problem specification. Defining and clarifying the terms and requirements he also introduced criteria to evaluate whether the desired objectives have been met.

In the justifying threads an issue is raised of what can be considered 'active' when referring to car suspension. These thoughts were important because they were recorded to allow following up any interpretations reached at the end of this episode. The initial analysis focused on two conceptual objects that can be 'active' or 'passive' – car chassis and shock absorbers. The designer initially framed the problem of active suspension in terms of a modifiable shock absorption and a constant chassis clearance. The initial frame led to the formal expression of goals listed in DC-II below.

Design episode T11.b (design context DC-I)

Development of DC-I and the emergence of basic concepts: The initial framing of the design problem led to the articulation of an 'ideal solution' (see notes J-495 and J-496 in Appendix B) that was formulated as a suspension system whereby a car driver would not feel or observe any unevenness of the road surface. This ideal state was further translated to the formal language of design as witnessed in the extract captured in justifications J-497 to J-502 in Appendix B.

At this stage, the designer defined basic conceptual objects of the respective design frame that he intended to use for a solution construction. For instance, a decision has been articulated that required a softer suspension for uneven surfaces and a tougher one for smooth roads (see J-497 in Appendix B). Another concept resulting from the initial framing was the monitor of the surface unevenness and the processing unit that would modify the shock absorber's behaviour (note J-498 in Appendix B). In addition to an automated response, the processing unit was required to respond to the explicit demands of the driver, hence the concept of 'manual adjustment' (note J-499 in Appendix B). Although the initial frame focused on the constant chassis clearance, the designer also articulated basic conceptual terms for adjusting the clearance. However, unlike the conceptual terms defined for the shock absorption, these were less precise (notes J-500 to J-502 in Appendix B).

Design episode T11.c (design context DC-I)

Retrieval of applicable components and the move towards the first principled sketches: At this stage, the designer expressed the concepts identified in the initial frame in a sketch (see Figure 7.2). The sketch showed how an absorber reacted to an uneven surface, and helped to introduce additional conceptual objects to the design frame. For example, a displacement of the wheel against the chassis was noted as a typical reaction to a car entering an uneven surface (denoted by parameter Δl). This observation was also formally noted in justification J-503 and refined in J-504 (see Appendix B). A new concept emerging from the sketch re-defined the generic concept of 'monitoring unevenness' (note J-498 in Appendix B); the new method of monitoring consisted of measuring the displacement of the wheel axle against the chassis.

Another concept that came out of the sketch was a helical spring acting as a familiar shock absorber prototype. However, despite the familiarity of the spring, it was not suitable for designing a sound solution. In several comments (notes J-498 and J-504 in Appendix B), the designer justified the decision not to use a spring model, because the shock absorber's toughness had to be modified in response to the state of the road. A spring as a mechanical component did not satisfy the required flexibility. Nevertheless, the spring as a prototype was instrumental in articulating the parameter to be amended in order to

change the shock absorption behaviour of the system. The parameter the designer became aware of after using a spring as a prototype was labelled as 'toughness constant K' and he started elaborating its influence on the shock elimination (see also the next episode).

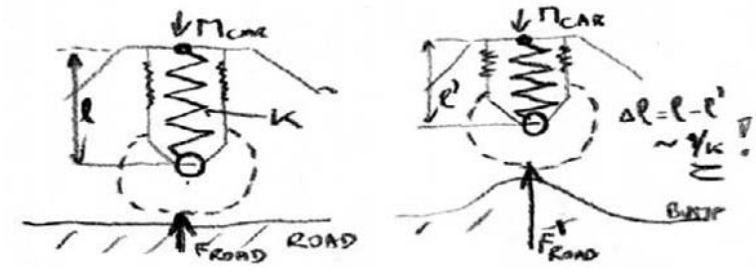


Figure 7.2. Behaviour of a prototypic shock absorber

Design episode T11.d (design context DC-II)

A new frame emerges 'informally': In the previous episode a new requirement was articulated in terms of amending the toughness constant K. Since in the spring, constant K depended only on its material, the designer looked for other models and materials, where this constant could be more easily modified. A frame containing similar conceptual behaviours as the one with a spring but different conceptual objects was brought forward in the form of pneumatic and hydraulic devices, pistons.

A simple pneumatic piston-based shock absorber was sketched (Figure 7.3) to replace the old spring-based prototype shown in Figure 7.2. The shock absorption behaviour of the piston depended on the type, volume and pressure of the substance inside. Since air pressure could be easily varied by pumping in more air or releasing it from the piston, the designer had a sound model for achieving the key objectives informally noted in frame DC-I. To acknowledge this progress, he recorded a new explicit frame and the corresponding new design context DC-II.

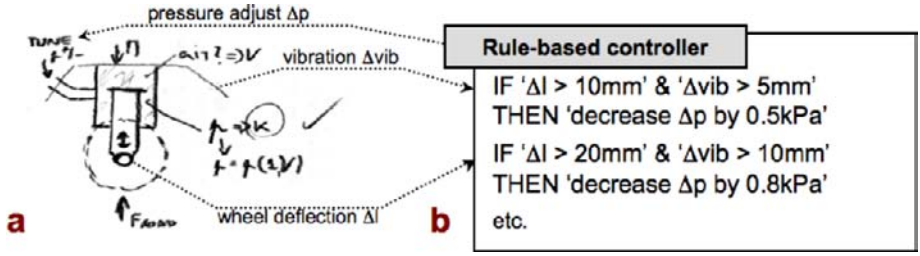


Figure 7.3. New prototypic device (a), and the first control loop solution (b)

Explicit achievement T11/DC-II

First explicit context with a solution is formally recorded (DC-II)[1]:

- the system should be able to absorb shocks (Req-1), and it must do so automatically depending on the current surface (Req-1.1);
- the system should adjust chassis clearance (Req-2), but for simplicity I assume constant clearance whilst driving (Req-2.2);
- the system should measure the displacement of the chassis against the road, or the wheels against the chassis (Req-3, Req-4);
- the system should be adjust dynamically the shock absorber's behaviour, especially its toughness parameter K (Req-5);
- the system should contain a processing unit able to decide how to react to a particular value of the wheel/chassis displacement and accordingly amend parameter K (Req-6)

The concepts for constructing the first solution arose in design episodes T11.a to T11.d, summarised above. The first milestone was accomplished by articulating solution Sol-1 (see also note J-514 in Appendix B) in the form of a rule-based control loop presented in Figure 7.3, sketch B. The behaviour of this controller was captured in a dedicated thread starting from note J-505 in Appendix B.

Design episode T11.e (design context DC-II, solution Sol-1)

Proposal of an exploratory probe in the new frame, re-addressing the concept of 'activity': From the notes following the articulation of solution Sol-1, (records J-515 to 517 in Appendix B), there was a visible

[1]All references to requirements and solutions are made with regard to data transcribed in Appendix A.

evidence that the first iteration was not acceptable in respect to the ideal state formulated earlier in note J-496 (see Appendix B). One of the designer's comments (see note J-515 in Appendix B) suggested that the reason for not accepting Sol-1 as a fully-fledged solution, was its inflexibility. The controller took into account only two inputs (displacements of wheels and chassis), and based on these, the air pressure was reduced to achieve a softer suspension. The system was unable to perform a reversed action, that is, reducing the pressure. The designer became aware of the fact that not only an uneven surface has an adverse impact on the driver's comfort, but so would have a too soft suspension; the car would start vibrating.

He proposed to address this discovered requirement by extending the simple control algorithm from context T11/DC-II (see notes J-516 and J-517 in Appendix B). The amended algorithm took into account the history of measurements done on the wheels and the chassis, so that when the initial shock had been eliminated, it started increasing the pressure automatically. The designer named this strategy as 'try to increment it & see it how far it is comfortable enough', which underlines its exploratory, heuristic nature. He also formulated new concepts useful in the prediction of the type of 'shock' (for example, a car entering an uneven road had a different ordering of events as a car entering a smooth road).

Design episode T11f (design context DC-II)

Another probe, re-addressing the chassis clearance (initial assumption of its constancy): In a similar manner to design episode T11.e, the designer was not satisfied with the restriction of a constant chassis clearance. It was helpful to produce the initial solution, but it clashed with his understanding of concept 'active suspension'. He addressed this inconsistency in a number of threaded records (see notes starting with J-518 and ending with J-527 in Appendix B).

The designer opted for a familiar prototype for controlling a parametric variable (in this case chassis clearance). First, he established a referential value for a stationary, empty vehicle. The displacement was measured after loading, and the individual wheels were appropriately balanced. Then a table was drawn up associating the measured shock on the chassis with the concept of 'terrain types'. To each terrain, a recommended chassis clearance was assigned, which was to be maintained by a controller.

New requirements (or re-specified and re-addressed requirements) triggered an articulation of new conceptual objects, from which sound solutions could be derived. One such new concept included a kind of 'screw-jack' (see Figure 7.4) that could be combined with the pneumatic piston known in the previous frame to implement an 'active suspension' (note J-521 in Appendix B).

Design episode T11.g (design context DC-II, DC-III)

Exploratory probes accepted, move to new context DC-III and new concepts for the implementation: Before implementing the two major extensions to context DC-II, the designer refined his description of the concepts that were known in that context. He focused on the articulation of physical and engineering principles behind the respective functional requirements and on refining abstract concepts from episodes T11.b to T11.d. The brevity of the records and the selected concepts reflect the designers decision to use the existing modules.

Thus, he proposed measuring chassis vibration and its displacement against the road surface using a technique similar to that for activating airbags (notes J-528, J-529 in Appendix B). For measuring the displacement of a piston from some initial position, he proposed a resistance-based measurement. A resistor became the means for measuring even diminutive changes in the incremental displacement (notes J-530 and J-531 in Appendix B). In the subsequent justifications (see notes J-532 and J-533 in Appendix B) he re-iterated the pressure modification in the pistons; he added an alternative of using oil instead of air (thus broadening previous frame and introducing hydraulic components). Figure 7.4 shows the parameters measured on the system labelled as 'IN'.

In addition to refining the concepts from DC-II, the designer proposed the same level of conceptual details for the emerged concept of the chassis clearance amendment. In this case, the prototypic incremental motors drove the screw-jack and move it up or down (notes J-536, J-537 in Appendix B). These motors are commonly used in controlling the motion in small devices.

Explicit achievement T11/DC-III

As it is possible to find in the explicit problem specification that served as a basis for this context, two statements appear refining

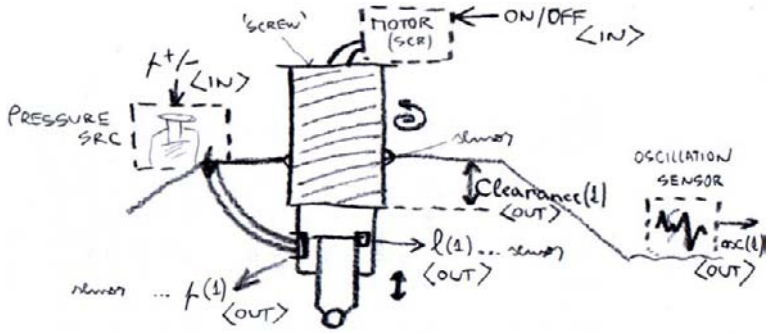


Figure 7.4. Extended candidate solution with new concepts and details

context T11/DC-II. In addition, two new requirements were added:

- the system should respond to both an immediate state of the road surface and explicit demand of the driver (Req-1.3) – refines (Req-1, Req-1.1);
- the system should adjust the chassis clearance when safety conditions allow (e.g., vehicle is stationary, steadily moving, not turning, braking, etc.) (Req-2.3) – refines (Req-2, Req-2.2);
- the system should contain a specific unit for controlling and adjusting the clearance, and also an 'action body' realising the adjustment (Req-7);
- it is necessary to co-ordinate the controls coming from the suspension controller and the clearance controller to ensure safe operation (Req-8)

To embody the above interpretation the designer constructed a new solution that contained the informally proposed new conceptual objects in addition to those brought in earlier. Solution Sol-2 in the decision sequence records served as a basis for further investigation as evidenced by the thread starting with record J-538 in Appendix B. A sketch of the two control loops, featuring also the concept of co-ordination is shown in Figure 7.5.

Design episode T11.h (design context DC-III)

Discovery and rectification of the unacceptable co-ordination: The solution in context T11/DC-III was constructed to remove the inadequacies of the previous iteration. As the follow-up to note J-538

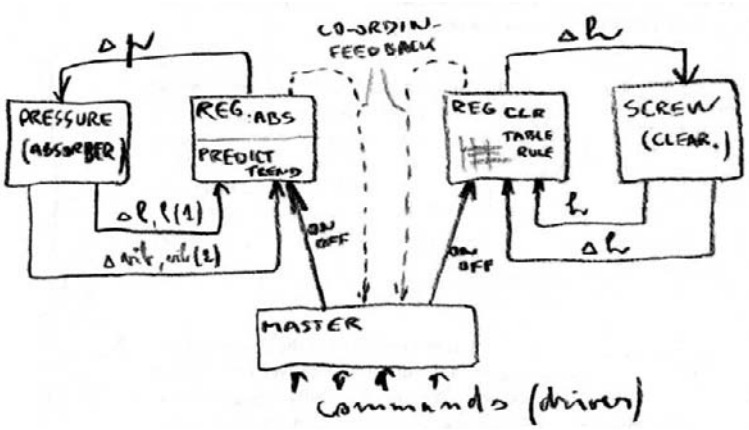


Figure 7.5. Another solution with a co-ordinating master controller

in Appendix B shows, the designer did not accept Sol-2 as the final design solution. He was not satisfied with the co-ordination of the two distinct control strategies.

The candidate solution Sol-2 was more complex than Sol-1, and included several modes of operation. The designer noted the emerging complexity in several notes, in which he made more specific commitments about the interactions between different modes, and about the interfaces conveying the information to the driver. For instance, he introduced concepts of driver overriding the automated control or turning off the clearance adjustment under specific conditions. These new requirements addressed the flaws of the current solution, and rendered the candidate solution Sol-2 unacceptable and inconsistent, thus requiring further elaboration and development.

Design episode T11.k (design context DC-III)

Consolidation of the extensions and refinements to the previous frames: The key concepts differentiating the new frame from the previous ones were 'interface' and 'man-system interaction' (noted in J-543, in Appendix B). The designer articulated the main functional requirement to be added in note J-544 in Appendix B. The functional purpose of an interface was to "translate the intuitive and qualitative values suitable for a driver to quantitative values recognisable by the suspension controllers". In the subsequent notes, the designer articulated

the concepts for implementing the new functional requirement.

First, he distinguished an 'overall mode' for choosing and switching between automated and manual adjustment of chassis clearance and of shock absorbing pistons. Another extension made a shift from quantitative controllers to fuzzy ones. The main benefit of this new controller was its capability to work with ranges that could be associated with qualitative terms, such as 'hard' or 'medium' for the suspension, and 'low' or 'very low' for chassis clearance.

As seen in the records J-545 and J-546 in Appendix B, this new conceptual approach improved also the individual controllers and the control algorithms. The rules were written more robustly and the overall behaviour became less prone to harmful vibration. For instance, the designer pointed out in notes J-516 and J-517 (Appendix B) that the new approach enabled better implementation of his original 'try & see' heuristic. Thus, formulating a new requirement and attending to it, he improved the system's performance also in respect to other issues.

Explicit achievement T11/DC-IV

After attending to various issues and resolving them, the designer proposed another refinement to the design solution. The only new requirement that emerged in the explicit specification in context DC-IV was regarding the interface:

- system should feature an interface for mediating between different modes of operation and driver's immediate choices (Req-9)

The solution associated with this particular context did not feature any major structural changes. The designer recorded it as Sol-3, and noted that it was similar to Sol-2 with a different internal implementation. He re-iterated that the most important addition would be a '(de)fuzzification' table associating the ranges of measured values (e.g., piston displacement, chassis oscillation, pressure, etc.) to and from the qualitative concepts (say, 'small', 'low', 'high', etc.), see also Figure 7.6.

Explicit achievement T11/DC-V

In addition to the construction of a new, robust solution Sol-3, the designer returned to a minor issue left from the early phases of de-

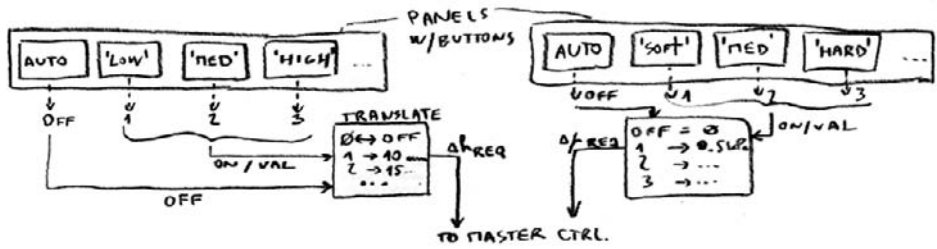


Figure 7.6. 'Interface' for communicating driver's requests

sign. He looked at the physical inputs and outputs of the controlled system, and their relationships. Once again, he tackled two issues while attending to the explicitly articulated problem of interactions between the controllers. In this last refinement to the solution, he formalised the interaction between the sub-modules responsible for the measurements. By doing so, he actually refined the previously vague articulations of the principles of measuring the desired values. The solution resulting from this last frame was encoded in the form of control flow diagram.

As the designer explained verbally, this was only a different recoding of the previous solution. He admitted that this form was better for establishing the interactions than the previous ones. He finished his task noting verbally that this schema was a reasonably complete and acceptable conceptual solution to the problem. Further work already required building a prototype. The physical outputs are in Figure 7.4 and the design session concluded with a solution.

7.3. RFD analysis of shock absorber design

I will next summarise each episode with a partial narrative of the design story in terms of the experimental objectives and the vocabulary of the RFD model from chapter 5. Each episode is associated with an appropriate design context and design frame. Some are clearly embedded in a single frame (for example, T11.c), whereas others serve as a transition from one frame to another. Typically, the second role appeared when the original frame was found insufficient and new conceptual terms were needed (for example, T11.b).

The narrative flow of the design story is interspersed by major

achievements that express the explicit commitments of the designer to a particular specification and all contain a solution associated with them. The designer recorded such milestones when he committed himself to conclusions that arose from the justifying threads. Thus, there is a visible relation between the two forms of talking about the same design problem. Explicit achievements represent separate design frames without any causal relations among them. The narratives serve as a lead capturing chains of decisions that shift the design from one frame to the next one. In some cases, the narratives are long (as between T11/DC-II and T11/DC-III), which suggests that the designer spent more time developing the next step by merging several discovered issues together. Other sequences of narratives are shorter, which suggests an immediate verification of the proposed resolution of the discovered conflict.

Design episode T11.a

Design episode T11.a initialises the problem solving space. It takes as an input the design brief, which can be labelled by symbol $\mathscr{DP}$. The reasoning steps performed by the designer were categorised as a retrieval of familiar frames (past design cases) that were similar to the current problem. The retrieval was followed by the articulation and selection of the relevant concepts that helped specify the current problem $\mathscr{DP}$. The purpose of design episode T11.a can be expressed by relation '$specifies(S, \mathscr{DP})$' from section 5.2.

In this session the following reasoning steps were observed: *action 2* and *action 3a* (defined in Table 7.1)

Design episode T11.b

The second chain of reasoning (design episode T11.b) can be seen as a development of the initial frame; it features a commitment to explicit specification S (goals, objectives of the session). However, unlike T11.a, this episode introduces the conceptual primitives to implement a solution satisfying the given specification. In the terms of my model, this chain explicitly articulates set $\mathscr{T}$ and the appropriate knowledge space $\mathscr{T}^*$ – a problem solving theory for this design frame. One can also note that some of the concepts were only introduced and others were elaborated into deeper details.

In this session, thus, the following reasoning steps occurred: *action 1a*, *action 3b*, *action 6a* and *action 6b* (referring to Table 7.1).

Design episode T11.c

Design episode T11.c further extends the approach started in T11.b. Unlike in the previous episode, first conceptual sketches appear at this stage – in the language of my theory, a potentially applicable problem solving model $T \subset \mathscr{T}^*$ is explicitly articulated. This articulation happened in terms of a familiar design frame (that is, choosing a spring as a body of the shock absorber). The commitment to this problem solving theory enabled further refinement of the problem specification S and of the conceptual foundation $\mathscr{T}$, and allowed the specification of controlled physical quantities. First time in the design process, the search for an admissible problem solving model $T \subset \mathscr{T}^*$ was observed, to satisfy the explicit problem specification developed so far (corresponding to predicate '$satisfies(T, S)$' in chapter 5).

In this episode, the following reasoning steps were noted: *action 2*, *action 3b*, *action 6b*, *action 5* and *4* (referring to Table 7.1).

Design episode T11.d

The informal proposal of a solution candidate in episode T11.c was followed by the assessment of its consistency. At this stage, the designer uncovered the first conflict, and addressed it by refining the specification of the problem (that is, demanding parametrised variables). He also brought in a new frame, a new familiar case. The frame introduced new conceptual terms at the level of structural primitives (set $\mathscr{T}$), such as a pneumatic piston.

In this episode, the following reasoning steps occurred: *action 6b*, *action 2*, *action 3a*, *action 3b* and *4* (referring to Table 7.1).

Explicit achievement T11/DC-II

Having informally assembled the pieces, the designer decided to make the first 'release' of a solution to the problem as he framed it. He confirmed his commitment to the informally noted requirements (Req-1 to Req-6 in T11/DC-II) and sketched a solution addressing them. In other words, he constructed a problem solving model T, which

satisfied these explicit requirements and proposed the first candidate solution that could be assessed for acceptability.

In this episode, the following reasoning steps were noted: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *5* (referring to Table 7.1).

Design episode T11.e

As the two subsequent episodes (T11.e and T11.f) prove, the candidate solution was not acceptable, and the designer modified his current interpretation of the problem. One part of the frame that showed potential for a modification was identified as inflexibility in the control algorithm. In order to fix this issue, the designer articulated a new requirement covering this incompleteness and extending his existing interpretation of the term 'active suspension' that emerged in one of the initial frames.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8*, *action 6a* and *5* (referring to Table 7.1).

Design episode T11.f

In a similar manner, another issue was brought forward in design episode T11.f, which eventually led to a modification of the initial assumption Req-2. Drawing on his past knowledge, the designer refined the original statement Req-2 to reflect the changed definition of term 'active suspension' from the previous episode. Thus, episodes T11.e and T11.f focused on the refinement and extension of the original problem specification ($S \rightarrow S' \subset \mathscr{S}^*$).

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8*, *action 2* and *3a* (referring to Table 7.1).

Design episode T11.g

Before validating these modifications of the original frame, the designer also tackled the issue of having too abstract conceptual primitives to work with (that is, sets $\mathscr{T}$ and $\mathscr{T}^*$). Therefore, in episode T11.g he elaborated some implementation details of the components known in the previous frame T11/DC-II, such as the measurement of piston displacement or the chassis clearance. Conceptually, he focused on refining the structural primitives (set $\mathscr{T}$ rather than extensions of the problem specification (set S).

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 4*, *action 2*, *action 3b* and *1b* (Table 7.1).

Explicit achievement T11/DC-III

Having found remedies to the identified inadequacies, the designer made the second explicit commitment to a candidate solution, that is, to satisfy the amended specification. In the transcript, this step is recorded as an explicit achievement T11/DC-III with a respective solution sketch Sol-2. As it happened with solution Sol-1, the acceptability of the proposed candidate had to be established.

In this episode, thus, the following reasoning steps were noted: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *5* (Table 7.1).

Design episode T11.h

Although more complete than Sol-1, as the design episode T11.h shows, candidate Sol-2 failed the acceptability test. The designer had to refine the current specification, and decided that the reason for failure was the lack of co-ordination between the two sub-controllers of the overall solution. Thus, a new requirement was added and a corresponding set of structural primitives was identified. In other words, this frame modification focused on the clarification of existing specification S and the extension of the conceptual set $\mathscr{T} \rightarrow \mathscr{T}'$.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8* and *action 6a* (Table 7.1).

Design episode T11.k

The concepts that emerged in design episode T11.h were further elaborated in T11.k translating the vague notions from T11.h into specific terms, such as 'interface' and 'mediation'. At the level of problem specification S, the purpose of the interface was expressed in terms of typical scenarios to be addressed. In the space of conceptual primitives (set $\mathscr{T}$), a fuzzy controller emerged in addition to the original control units, in order to mediate the driver's qualitative requests for change.

In this episode, thus, the following reasoning steps were noted: *action 2*, *action 3b*, *action 4* and *action 5* (Table 7.1).

Explicit achievement T11/DC-IV

The proposed modification of the frame used in context DC-III was verified by an explicit commitment to the new requirement and construction of another solution (Sol-3). This sequence is recorded as an explicit achievement T11/DC-IV. The new problem solving model ($T \subset \mathscr{T}^*$) was articulated so that it satisfied the modified problem specification S and was assessed for acceptability.

In this episode, thus, the following reasoning steps were noted: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (Table 7.1).

Explicit achievement T11/DC-V

The acceptability test did not pass unconditionally, because the designer made a small amendment to frame DC-IV before declaring a design solution. However, as record T11/DC-V shows, this was not so much the failure of the acceptability test as the completeness and clarity test. This reflective step triggered minor refinements of the structural primitives, and, in turn, seeded a refined conceptual frame in the new explicit context DC-V.

In this episode, thus, the following reasoning steps were noted: *action 4*, *action 7*, *action 6b* and *action 5* (Table 7.1).

Table 7.2, gives a more concise account of our interpretation of the recorded and annotated design sequence. For clarity, the interpretation is conducted using the vocabulary from chapter 5. The legend associates the referential numbers of reasoning steps with the verbal descriptions. The columns correspond to the individual design episodes of the narrative as described earlier in section 7.2. The values in the individual cells reflect the order in which the different actions were observed.

7.4. Annotated task T21 – paper-smoothing plant

The initial design brief given to the designer reads as follows: *Paper-milling plant consists of several different chains of machinery that produce different types of paper. We are interested in one particular plant that takes semi-finished paper on a roll as an input, smoothes it, and rewinds it on another roll on output. Your task is to design a part*

Table 7.2. Conceptual summary of task T11 with design episodes 8.2a to 8.2k

Step	a	b	c	d	DC-2	e	f	g	DC-3	h	k	DC-4	DC-5
1a		1.			1.				1.			1.	
1b					1.			4.	1.			1.	
2	1.		1.	2.			3.	2.			1.		
3a	2.			3.			4.						
3b		1.	2.	3.				3.			2.		
4			4.	4.				1.			3.		1.
5			3.		3.	4.			3.		3.	3.	3.
6a		2.			2.	3.			2.	3.		2.	
6b		2.	2.	1.	2.				2.			2.	2.
7						1.	1.	1.		1.			1.
8						2.	2.			2.			

of a paper mill to perform and control the above-defined operation, and to ensure that the paper is evenly thick, smooth, and wound tightly on the output. The raw paper at the input may have ripples, variable thickness, rough surface, or any combination of defects. In addition to designing the plant, suggest also a robust control strategy, particularly with regard to paper thickness and strength as key control parameters.

Design episode T21.a (design context DC-I)

Initialisation of the first frame in context DC-I: The designer used the design brief shown above and conducted an initial task analysis to clarify the objectives and to specify the problem in a formal language. The introduction was captured in justification note J-692, in which the designer sketched how an input and output to/from the designed system might look like. Based on the initial drawing, he started articulating the parameters describing the problem, and expressed the desired functionality of the designed system in notes J-693 to J-695 (see Appendix B).

While the reference to the past experience is observable in the captured notes, it is rather subtle. The designer used a familiar vocabulary, such as 'reducing paper thickness', 'polishing paper surface' to articulate the initial frame for the interpretation of the problem. Familiar vocabulary also helped him in note J-695 (in Appendix B) to attend to the issue of how much the paper thickness may be modified by the system. This initial frame led to the formal expression of the task objectives, explicitly articulated later in context DC-II.

Design episode T21.b (design context DC-I)

Development of the initial frame and the emergence of explicit objectives: In addition to the clarification of the concepts from episode T21.a, the designer brought forward additional parameters and requirements. As can be seen from the notes in Appendix B, these originated in his generic knowledge of the control theory and experience with controlling mechanical systems (J-696). For example, he demanded additional restrictions from the customer, and enquired about the typical dimensions and weight of the processed paper (notes J-697, J-699).

Another issue that was not mentioned in the initial brief but was attended to during the refinement of the initial frame was the expected initialisation of the process. The designer pointed out that the system might require manual start-up, for instance, to feed the paper through the assembly onto the output roll (see note J-703 in Appendix B). This assumption did not lead to any frame amendment, however. More important consequence was that a similar situation might occur at the end of the process. Thus, he articulated the need to monitor when the paper at the input roll is running out, so that the system could halt in time before tearing the paper from the input roll (notes J-703 and J-704 in Appendix B).

Explicit achievement T21/DC-II

Despite rather abstract definition of the desired parameters of the ideal solution that appeared in episodes T21.a and T21.b, the designer made an early explicit commitment to the following goals and requirements he aimed to achieve:

- the system should be able to remove ripples from the raw paper on input (Req-1);
- the system should also be able to polish the surface so that it does not contain any rough debris (Req-2);
- the system should be able to reduce thickness to the value specified by the operator, so that paper is evenly thick throughout (Req-3);
- finally, the controller should be able to detect the paper on the input roll running out and stop the process before the paper is abruptly torn off the input roll (Req-4)

Since the initial frame was articulated rather abstractly, it did not contain the necessary conceptual primitives to construct a technologically feasible solution. Therefore, the designer used a 'black box' approach to express the solution functionally. Such approach enabled him to work with an early candidate solution without committing himself to any particular technology. The system in this abstract solution Sol-1 is referred to as 'smoothing unit' and the designed system also contains an 'input' and 'output rolls' (see Figure 7.7).

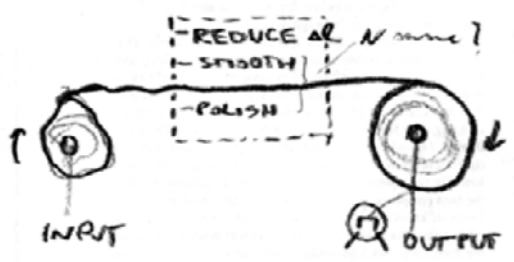


Figure 7.7. Initial abstract formulation of the design problem

Design episode T21.c (design context DC-II)

Extension of the initial frame and details of the 'paper smoothing unit': The solution sketched within frame DC-II needed further refinement before any acceptability could be tested. Therefore, the designer investigated the means how the desired objectives could be achieved and one technology that appeared in the justifying records drew upon the idea of rolling and the application of drum pressure (see notes J-708 and J-709 in Appendix B). The designer sketched several implementations of this technology, for example, a single drum on 'a table' or a pair of drums with a gap in between. Some of these alternatives are shown in Figure 7.8.

Rolling was a familiar technology that was applicable and could be re-used in this particular problem. However, the designer did not accept it off the shelf. Record J-709 shows a quick acceptability check and the identification of potential issues – the main concern was that the paper might be damaged when rolled and the operation might not be efficient. From the experience he recalled that rolling ('ironing')

was often accompanied by some form of pre-processing, for instance, material 'warming up', 'wetting', or 'melting'.

Therefore, the designer introduced an additional requirement to the problem specification assuming that the paper would be treated before entering the rolling assembly (see note J-711 in Appendix B). This requirement was to ensure efficiency and reduce the danger of paper damage that might have occurred if the pressure on a dry paper was increased. Simultaneously with the new assumptions, the concepts for constructing the detailed solutions emerged, and the designer proposed how such a moisturiser might look like.

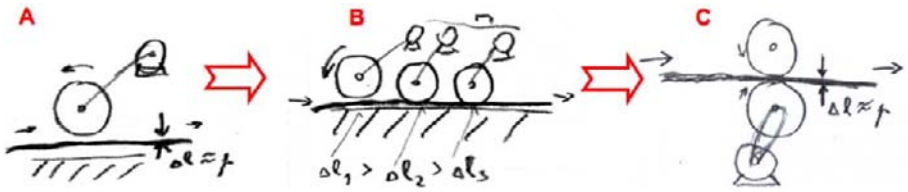


Figure 7.8. Functional alternatives addressing the explicit requirements

Design episode T21.d (design context DC-II)

Further refinements of the frame and ordering of the operations: Following the proposals made in episode T21.c, the designer elaborated the details of the newly articulated concepts of 'wetting' and 'rolling' (J-717). One important concept emerging from his commitment to this frame was an order, in which the operations take place and consequently a layout of the designed system. Since there was a clear causal dependency between wetting and rolling, the order was given by this causality – first 'moisturise', then 'apply pressure between the rolling drums' (J-718). To keep the conceptual solution simple the designer chose a single pair of drums with a gap corresponding to the desired thickness (J-719).

This explicit attendance to the details of the interactions between the operations introduced the need of another operation. After leaving the rolling drums, the paper was still damp, and as such, it could not be stored on the output roll. Therefore, the designer rectified this inconsistency by bringing in a unit to dry the damp paper. In

other words, another requirement emerged, and the corresponding structural concepts were identified (J-720 in Appendix B).

Explicit achievement T21/DC-III

As noted in episodes T21.c and T21.d, several new requirements emerged during the refinement of the prototypic solution Sol-1. In addition to those explicitly noted in context DC-II, the extended context DC-III contained the designer's explicit commitment to the following three objectives concerning paper pre-processing:

- the paper should be treated before any manipulation by moisturising the dry paper delivered to the input (Req-5);
- once the paper is wetted, ripples should be removed and thickness reduced (Req-6), this is also addressed partially by Req-1 and Req-3;
- the control should ensure that paper at the output is dry; a mechanism removing excessive moisture is needed after the smoothing unit (Req-7)

The newly-articulated objectives were embodied in a more realistic, though still rather simple solution. The designer implemented the moisturiser by 'shower-like jets' releasing warm damp. Rolling was confirmed as a prototype for smoothing, and finally, drying was implemented as 'a tunnel' with hot, dry air where the excessive moisture evaporated. Warm paper leaving the drying tunnel was immediately wound on the output so that it 'did not wrinkle' as an effect of drying. This solution was labelled as Sol-2 and is sketched in Figure 7.9 (see also context T21/DC-IV).

Design episode T21.e (design context DC-III)

Exploratory probe addressing the issue of thickness reduction: Using a similar strategy as described in explicit achievement T21/DC-II the designer addressed another outstanding issue with thickness reduction. In note J-723 (in Appendix B) he identified the reason of his dissatisfaction with solution Sol-2 as having insufficient means for thickness reduction. Although the rolling unit could be re-used for thickness reduction, the single pair of drums had undesired limitations on the degree of thickness reduction; that is, how much can be reduced by a single drum pair (note J-725 in Appendix B).

The designer demanded a more flexible approach that would reduce thickness to a greater extent. To address this refined requirement of thickness reduction, he articulated a new concept of a sequence of rolling drums in J-726. Another concept emerging from assessing the acceptability of solution Sol-2 featured decreasing values of the gap clearance between the rolling drums (noted in J-727 in Appendix B). Starting with the first pair (closest to the moisturiser) the gap clearance would decrease, so that the last pair of drums had the desired thickness. This approach would reduce the thickness step-by-step thus avoiding potential dangers of clogging the rolling assembly with excessive paper left-overs.

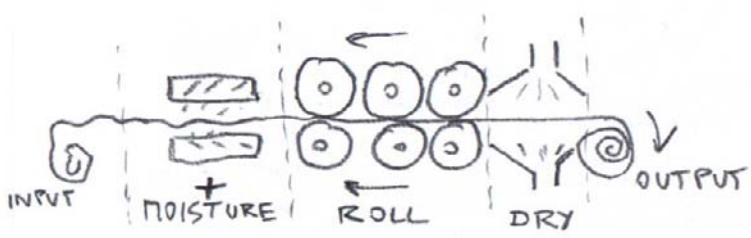


Figure 7.9. Simple solution extended and refined

Explicit achievement T21/DC-IV

After introducing new conceptual terms to address inadequate behaviour of the previous solutions (Sol-2) in episode T21.e, the designer recorded the explicit commitment to merging the 'smoothing mechanism' with the 'thickness reducing mechanism'. He implemented them as one rolling unit, and the refined specification of the problem contained the following statement:

- the desired purpose of the smoothing unit should be co-ordinated with the thickness reduction; for instance, by multiplying the number of smoothing units each having a different gap clearance (Req-6.1 extending original Req-6 from DC-III)

The new frame in context DC-IV did not differ from the previous one in many aspects. The most striking difference was the introduction of a sequence of rolling drums and their linear arrangement with a decreasing gap clearance between the drums in each pair. The

sequence replaced the single pair from solution Sol-2, and being conceptually distinct, it was, therefore, labelled as Sol-3 (depicted as a sketch in Figure 7.9).

Design episode T21.f (design context DC-IV)

Re-addressing the issue of thickness reduction and paper damage: With Sol-3 being a realistic candidate solution to the specified problem, the designer assessed its acceptability, and became aware of several drawbacks. As record J-731 in Appendix B suggests, the issues likely arose after sketching the concept of 'a sequence' in the designer's notebook. His main concern is visible when comparing Figure 7.8c with the sketch in Figure 7.9. In the note the designer suggested that such a layout might be large and impractical to be controlled. Moreover, the damp paper could still be torn in the rollers when pulled through, which was unacceptable and contradicted the explicit requirements.

As record J-733 in Appendix B shows, he first considered a simple remedy by 'squeezing' the drums; that is, placing them closer to each other, thus reducing the overall length of the assembly. However, he was not satisfied with this fix and explored other opportunities. The concept of 'squeezing' helped to trigger an analogy, because the next record (J-734) shows the emergence of an entirely new conceptual term. The new concept emerged first in the language of existing frame, where it required 'squeezing the drums in two dimensions' instead of one. This cumbersome definition was followed up and reformulated in note J-736 in Appendix B.

Thus, a new concept of 'an alternate drum layout' was introduced in J-736 as an alternative to 'the linear layout'. This idea was elaborated in J-738 and proved to be promising, because it featured much larger 'effective surface' acting on the paper (another new concept emerging from the frame extension). Thanks to a more efficient rolling, the pressure of the drums could have been reduced and the gap clearance ceased to be a major restrictive factor.

Explicit achievement T21/DC-V

The newly-emerged alternative for re-arranging the drums was formally recorded as a required assumption that replaced the old Req-6.1 from context DC-IV:

- the functionality of smoothing should be co-ordinated with the need of thickness reduction; for instance, by re-arranging the smoothing units in an alternate (zigzag) manner Req-6.2

The plant featuring the alternate layout of the rolling units and their control was recorded as solution Sol-4, and it is depicted in Figure 7.10. As it was observed in T21/DC-III, the two alternatives are functionally similar but conceptually very different. They are based on different conceptual primitives and physical principles.

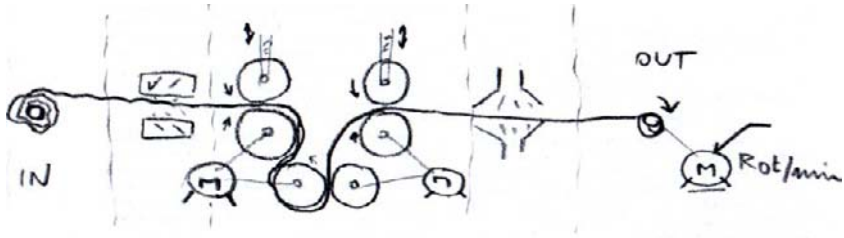


Figure 7.10. Introduction of a concept of alternately positioned rolling drums

Design episode T21.g (design context DC-V)

Currently controlled devices (motors, rolls) incompatible with the new layout: The next step appearing in the recorded notes transferred the design focus to the actual control strategy (noted in J-754 in Appendix b). So far, the control was simple – adjusting all the parameters at the beginning and monitoring only the end of paper at the input roll (see also episode T21.b). However, such strategy did not suit the latest layout of the assembly, and needed to be re-addressed. The first point was clarified in terms of thickness being constant for a given input; that is, it would not change during the smoothing operation (note J-755).

This new assumption helped to articulate the control strategy based on monitoring the input to the smoothing unit (J-756 in Appendix B). The parameters measured at the input would apply throughout the process; including approximate diameter and weight of the roll. Similarly, the amount of water or damp sprayed on the paper in the moisturiser would be determined by the initial humidity of paper. The control action acted upon the output roll (noted in J-759) that

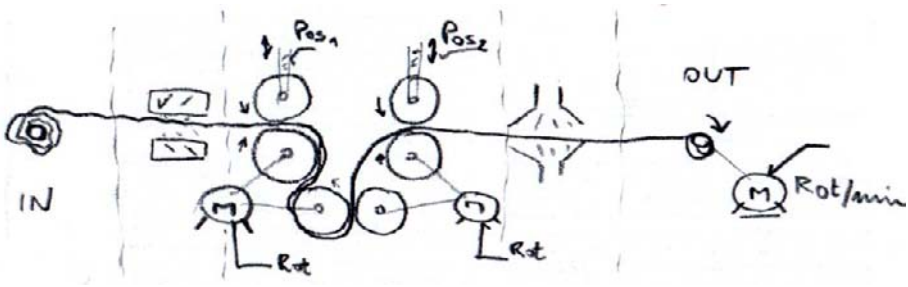


Figure 7.11. Smoothing plant with alternate rollers and control parameters

was coupled with a motor that rotated the roll with an appropriate velocity. The velocity corresponded to the actual diameter of the output roll (which was also measured), and was controlled so as to ensure certain tension of the paper throughout the smoothing assembly.

The designer started with a simple control that drove the output roll (note J-762 in Appendix B). This drive was simultaneously responsible for unwinding the paper from the input roll and pulling it through the moisturising unit, rolling unit and drying unit. Another functionality required from the control was identified as maintaining the tension of paper thus avoiding the undesired wrinkling.

Explicit achievement T21/DC-VI

To address the issues from design episode T21.g and to further develop the previous solution Sol-4, the designer articulated two additional requirements. Unlike the requirements recorded in the previous frames, these were concerned with the details of the control strategy rather than with the smoothing function. New context DC-VI was articulated and contained these two requirements:

- first, a mechanism was needed for unwinding the paper from the input roll and simultaneously winding it at the output (Req-10);
- second, an adjustable driving force was required to rotate the output roll, thus maintaining the paper tension (Req-11)

A solution articulated to address the newly formulated needs was labelled as Sol-5 and, in principle, it was structurally similar to Sol-4. However, as Figure 7.11 shows, it contained additional concepts, such as 'sensors measuring roll diameter at the input' (R_{IN}), 'humidity

sensor' (Hum_I), gap adjusters (Pos_1, Pos_2), controlled parameter (Rot/min) and an appropriate actuator at the output (motor). These new conceptual objects were positioned at the appropriate places of the design sketch.

Design episode T21.h (design context DC-VI)

Conflict between the control strategy and damp paper: However, after sketching the proposed drive of the output roll, the designer discovered contradictory requirements that increased the danger of damaging paper (see note J-766 in Appendix B). On one hand, the output roll torque was maximal at the beginning of the process when most of the paper was at the input. On the other hand, paper was wetted and fed into the complex rolling assembly. Driving the output roll meant transferring potentially large torque exerted at the output through the entire system, so that the input roll started unwinding. For thinner and damper paper, this could tear the paper, which was unacceptable.

Therefore, the designer amended the requirement of a single adjustable drive, and proposed controlling both input and output rolls based on the paper tension measurements between them (note J-767 in Appendix B). Thus, he modified the original concept of 'paper pulling' from frame DC-VI to a combined 'pulling of the output roll' and 'pushing/feeding' coming from the driven input roll. The proposal balanced potentially dangerous torque and spread it evenly at both ends of the designed plant.

Design episode T21.k (design context DC-VI)

Refinement of the new conceptual interpretation of input/output control: Before implementing the deliberation from episode T21.h explicitly, the designer assessed the acceptability of the latest idea. As record J-768 in Appendix B shows, the 'pushed paper' could cause problems similar to those identified earlier in design episode T21.e. The rolling drums could be clogged if the input roll pushed the paper too hard. The danger was emphasised by the fact that the system was feeding wet paper, which was more difficult than feeding the dry paper, as it did not keep the shape.

Therefore, the designer returned to the concept of 'pulling paper' from the input roll. However, this time, a revised version was pro-

posed in note J-769. It assumed driving an additional device located between the input roll and the moisturising jets. It was suggested that this extra unit consisted of two 'drums' driven so that the paper tension is maintained on a constant level. A similar pattern was re-applied at the output, and this remedy solved the problem of clogging the rollers, because only that much paper was unwound as to maintain the tension.

Design episode T21.l (design context DC-VI, DC-VII)

Following the introduction of a new conceptual object: A device pulling paper from the input roll and ensuring its tension triggered the proposed of an interesting re-shape of the plant. This was not recorded verbally but emerged in one of the last drawings. The idea from the initial frame T21/DC-II of using pairs of rolling drums was radically revised, and instead of pairs, single drums were arranged in a zigzag fashion. As the designer noted, this was sufficient and simpler than controlling complex drum pairs from Figure 7.11.

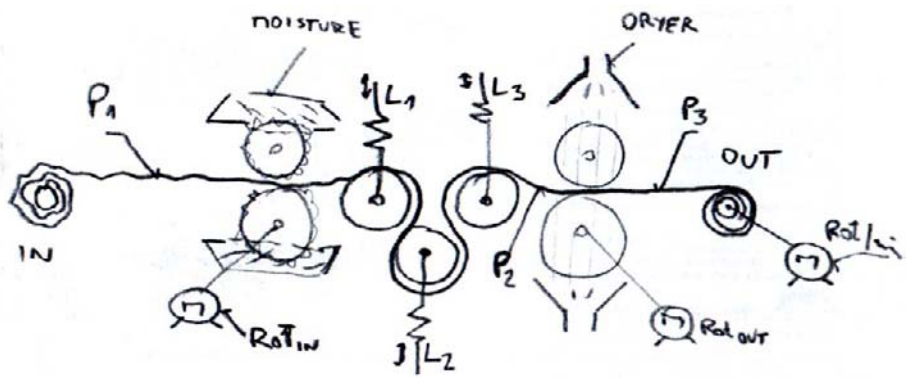


Figure 7.12. Re-shaped solution addressing the re-framed problem

From his description of the final layout one observes the return to the concept of 'polishing' that already appeared in episode T21.a and context T21/DC-II. The same drums that were originally applying pressure on the paper were also generating friction. Using this interesting collateral effect, the principle of rolling was re-interpreted from the 'pressure application' to 'pressure and friction based abrasion'. The latest iteration in the solution development with the new layout of rollers is shown in Figure 7.12.

Explicit achievement T21/DC-VII (DC-VIII)

As captured in the sequence of design episodes T21.h to T21.l, the specification of paper smoothing problem was extended so that there were two drives. This new requirement extended an earlier requirement Req-10 from frame DC-VI. In addition to refining Req-10 and as its consequence, a requirement similar to Req-11 from frame DC-VI was introduced. Below is the list of the extensions forming new frame DC-VII:

- first, a mechanism was needed for unwinding paper from the input roll and another (physically independent but co-ordinated) one was needed for winding paper at the output (Req-10.1);
- second, an adjustable driving force was needed to unwind paper from the input roll that would simultaneously maintain the tension at the input and prevent clogging (Req-12)

As mentioned in episode T21.l, the designer eventually re-shaped the layout of the plant to reflect the latest additions to the problem specification. The re-shaping followed the implementation of the component addressing requirement Req-10.1, and thanks to the designer's re-interpretation of concept 'polishing'. The result of the design process, proclaimed a design solution, is depicted in Figure 7.12, and a rough schema of the control loop with the measured and controlled parameters appears in Figure 7.13. Hereby the design session ended.

7.5. RFD analysis of paper-smoothing plant

Let me summarise the annotated transcript of session T21 in terms of the RFD model introduced in chapter 5. The captured records revealed that task T21 was tackled in seven successive conceptual contexts. In the episodic narratives in section 7.4, these are labelled from DC-I to DC-VII. There were ten design episodes labelled from T21.a to T21.l. A shallow conclusion reveals a similar number of self-contained reasoning chains as observed in task T11 (section 7.3). However, the pattern of the overall design process is different.

First, there are more explicit articulations of the partial solutions than it was in task T11, and the number of explicit milestones (T21/DC-II to DC-VII) is greater – six contexts in T21 compared to four in T11. Second, the distribution of the narratives and the ex-

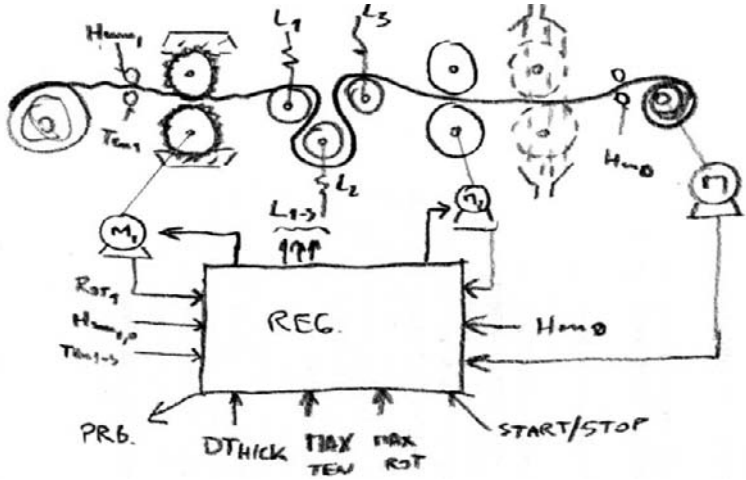


Figure 7.13. Control loop with respective parameters extending Figure 7.12

plicit achievements is also different. These two differences suggest a more structured, more exploratory approach to the design of a paper-smoothing plant; the designer explored and constructed the design space by articulating small probes, which he immediately attempted to apply to validate their feasibility.

The pattern of structuring the design problem is observable in the interweaving of the explicit specification extensions and the solution articulations (T21/DC-III to DC-VI), and also in the probing extensions to the problem interpretation (design episodes T21.e to T21.g). Several of these exploratory probes contain frame amendments, which supports my hypothesis from chapter 5 about a frequent mutual interaction between the problem specifications and solutions. Similarly as in section 7.3, shorter sequences of design episodes are related to the immediate validation of the amended frames.

Design episode T21.a

Task T21 started with the population of the initial context DC-I from the design brief, and the explicit specification of the problem ($\mathscr{D}\mathscr{P}$). The process of framing featured a retrieval and application of the familiar past cases and the re-use of their vocabulary for the specification of the main objectives in episode T21.a. Little commitment was made to the articulation of structural primitives; in the

first episode, the designer focused on the basic tenets of the explicit problem specification. Rather than labelling these notions as set S, they exhibited more generic features of the conceptual specification set $\mathscr{S}$ (see chapter 5 for the differences).

In this episode, thus, the following reasoning steps were noted: *action 1a*, *action 2*, and *action 3a* (cf. Table 7.1).

Design episode T21.b

The identification and selection of the requirements from set $\mathscr{S}$ defined in episode T21.a, that is, the articulation of an explicit problem specification $S \subset \mathscr{S}^*$, paved the way for the identification of basic structural concepts (set $\mathscr{T}$) in episode T21.b. These concepts included the start-up of the process, monitoring paper running out, and measuring weight, diameter. Whether a particular familiar term was included in the initial frame of the design problem depended on its admissibility. The designer performed a simple check on consistency of the familiar objects with the existing goals of the explicit problem specification S.

In this episode, thus, the following reasoning steps occurred: *action 2*, *action 3b*, *action 6a*, *action 6b* and *action 4* (cf. Table 7.1).

Explicit achievement T21/DC-II

At this stage, the designer presented the first principled solution, in which the basic conceptual terms were deployed. These concepts were later refined and instantiated using a more specific terminology. While the proposed solution $T \subset \mathscr{T}^*$ was too abstract for the purposes of implementation, it provided a useful referential framework for the subsequent iterations. In the language of the RFD model, the designer proposed a problem solving model for the explicit problem specification; that is, formed predicate $satisfies(T, S)$.

In this episode, thus, the following reasoning steps occured: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.c

Contrary to my hypothesis from chapter 5, the designer's behaviour in episode T21.c does not show a clear assessment of the proposed solution model for its acceptability. Instead, he continued with the

development of the frame started in episode T21.b. This pattern, however, supports the explanation from the beginning of this section about the exploratory probing. In other words, the acceptability test was incorporated in upholding of the explicit specification ($S \subset \mathscr{S}^*$) and the problem solving theory (set $\mathscr{T}^*$) during the construction of the prototypic solution Sol-1. Thus, the designer refined the conceptual primitives for the construction of a more specific solution; he added the rolling drums and technology of rolling. He also introduced new concepts from his experience that were applicable to the problem $\mathscr{DP}$; e.g., the requirements to operate efficiently and avoid damage to the paper.

In this episode, thus, the following reasoning steps were noted: *action 4*, *action 7*, *action 6b*, *action 2* and *action 3a* (cf. Table 7.1).

Design episode T21.d

The vaguely identified extension of the current conceptual frame and problem specification started in episode T21.c continued also in T21.d with further refinements and commitments. The designer extended the conceptual set $\mathscr{T}$ with such terms as 'wetting' and 'drying'. Another contribution made in this episode was the introduction of ordering between the operations realising paper smoothing, that is, 'unwinding', 'wetting', 'rolling', 'drying' and 'winding'. The order of operations was found important, and the designer incorporated an explicit requirement for an order into his explicit problem specification S.

In this episode, thus, the following reasoning steps were noted: *action 3b*, *action 6a*, and *action 6b* (cf. Table 7.1).

Explicit achievement T21/DC-III

Since several new requirements emerged since the previous solution was constructed, the designer made another commitment to a partial solution satisfying the amended problem specification. In terms of my model, explicit record T21/DC-III corresponds to the construction of a new set $T' \subset \mathscr{T}^*$, for which predicate '$satisfies(T', S)$' would hold. Such a problem solving model was noted by milestone T21/DC-III in the iterative design of a paper-smoothing plant.

In this episode, thus, the following reasoning steps occurred: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.e

Unlike the previous solution developed in context T21/DC-II, solution Sol-2 was more specific and it could be assessed for its acceptability. As documented in episode T21.e, the designer did not accept this particular implementation, and re-visited the requirement demanding thickness reduction. In order to articulate the missing parts, he brought in several new assumptions about the flexibility of the thickness reduction, as well as the requirement of a co-ordination between paper smoothing and thickness reduction. In the language of my theory, design episode T21.e corresponds to the refinement of the problem specification ($S' = S \cup s$) and features a frame development.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8*, *action 6a* and *action 5* (cf. Table 7.1).

Explicit achievement T21/DC-IV

The extensions proposed in design episode T21.e were immediately tested by a proposal of another candidate solution (Sol-3) as a part of context T21/DC-IV. This context focused on the co-ordination between the two operations as demanded in the modified problem specification in T21.e. The designer looked for another problem solving model ($T'' \subset \mathscr{T}^*$) to satisfy such a specification. New model was found, and it contained a few new conceptual terms that appeared informally in episode T21.e (e.g., a sequence of the rolling drums, decreasing gap clearance, and stepwise reduction of thickness). The solution with a linear sequence of drums addressed the explicit problem specification, but needed to be assessed for its acceptability.

In this episode, thus, the following reasoning steps occurred: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.f

The acceptability assessment occurred in episode T21.f, where the designer reflected on the sketch made as a part of solution Sol-3. Although as a first approximation the solution was sound, the designer became aware of several issues with the proposed layout. One problem was identified as impractical length of the assembly that might pose difficulties with control and maintenance. Another issue that contradicted his explicit problem specification was concerned about

tearing the wet paper when too much pressure was applied. Thus, the co-ordination of smoothing and thickness reduction had to be re-addressed, and the unacceptability of paper tearing was re-iterated. In addition to an extended problem specification, a new concept of two-dimensional (zigzag) arrangement of drums emerged leading to an extension of conceptual base $\mathscr{T}$.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8*, *action 2*, *action 3a*, *action 3b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Explicit achievement T21/DC-V

The re-formulation of one requirement and the respective solution was immediately tested by sketching it in the notebook (see T21/DC-V). Though functionally the proposed solution Sol-4 behaved similarly is Sol-3 or Sol-2, it was conceptually different. It introduced the zigzag layout of the drums that survived until the end of the design process. Another contribution of this explicit commitment is a clear presence of an iterative development of a requirement; for example, the requirement Req-6 was re-formulated twice since its conception.

In this episode, the following reasoning steps were noted: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.g

Structurally, proposal Sol-4 was found acceptable; however, the simplistic control strategy used from the beginning was not suitable for a more complex structure. Therefore, in episode T21.g the designer re-focused his objectives to amend this strategy. He decided to control the output roll, and the influential parameters were measured at the input of the plant. To express his preference about the control strategy, he articulated a requirement that the output roll be responsible for winding the paper at the output, unwinding it from the input and pulling it through the plant. This new statement extended the problem specification (S') and was complemented by an articulation of new conceptual objects for its implementation, sensors and motors (extending set $\mathscr{T} \rightarrow \mathscr{T}'$).

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 4*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Explicit achievement T21/DC-VI

As in the previous episodes, new requirements were implemented and the designer recorded another explicit commitment to the proposed structure and control strategy, as shown in the explicit achievement T21/DC-VI and respective solution Sol-5. The focus shift was manifested by an introduction of new requirements to the problem specification ($S'' = S' \cup \{s\}$). These were formulated to address the outstanding issues with the previous solution Sol-4, which needed to be amended as well. The new solution model perused the newly introduced conceptual primitives (see set $\mathscr{T}'$ in design episode T21.g), and the essential contribution of this episode was their incorporation into existing structure of the previous solution Sol-4.

In this episode, thus, the following reasoning steps occurred: *action 1a*, *action 1b*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.h

The exploratory probe conducted in T21/DC-VI revealed an issue with exerting too large torque and tension on the paper when pulling it through the entire assembly as proposed in Sol-5. Design episode T21.h returned to the control strategy and modified the preference for a single controlled point (the output roll) thus re-formulating the explicit problem specification. As a quick fix to the issue, the designer proposed driving and controlling both input and output rolls, thus pushing and pulling the paper. The aim of this shift was to distribute the torque and tension more evenly.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 8*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.k

As design episode T21.k documents, the tentative proposal of pushing (feeding) the paper into the rolling assembly did not pass the acceptability check. The designer resolved that it was not a feasible alternative, because in certain instances, it could make the issue of damaging the paper worse. Therefore, he proposed a new conceptual primitive to address the conflict within paper feeding. In the language of the theory from chapter 5, he amended both sets forming a

design frame, $\mathscr{S}$ and $\mathscr{T}$. However, it is difficult to say in what order did this happen.

In this episode, thus, the following reasoning steps were noted: *action 7*, *action 6a*, *action 6b* and *action 5* (cf. Table 7.1).

Design episode T21.l

Before testing these probes in an explicit solution, the designer proposed an interesting alternative to the existing structure of the designed plant. Design episode T21.l does not contain any explicit reference to an unacceptable behaviour; however, it features a significant frame shift. In my opinion, the acceptability test was conducted internally, and the designer believed he could further improve his design although not being able to put his finger on any explicit conflict. What happened in this last occurrence of re-framing re-interpreted the functionality of existing structures (rolling drums). This reasoning sequence shows very nicely the principle of the conceptual reformulation rather than extension or refinement that were observed in the previous episodes. The conceptual base $\mathscr{T}$ remained the same as in context DC-VI; however, the meaning and purpose of the individual elements changed (see the shift from 'pressure application' to 'abrasion' in episode T21.l).

In this episode, thus, the following reasoning steps were noted: *action 8*, *action 3a*, *action 3b* and *action 5* (cf. Table 7.1).

Explicit achievement T21/DC-VII

Eventually, the probes proposed in design episodes T21.h to T21.l were incorporated into an explicit account of a solution that was finally found reasonably complete and acceptable to address the given design problem $\mathscr{DP}$. The set of requirements recorded in context T21/DC-VII sufficiently specified the problem $\mathscr{DP}$, and the last proposed alternative of a solution (Sol-6) satisfied this problem specification. In addition to the satisfiability, solution Sol-6 was also tacitly acceptable.

In this episode, the following reasoning steps occurred: *action 1a*, *action 1b*, *action 6a*, *action 6b*, *action 5* and *action 8* (cf. Table 7.1).

Table 7.3 gives a concise account of the recorded sequence of design episodes. For clarity, the interpretation is conducted using the vo-

cabulary from chapter 5. The columns in the table correspond to the individual design episodes of the narrative as described in section 7.4 (ranging from [T21]a to l). The values in the individual cells reflect the order in which the different reasoning steps were observed – if such ordering was possible to be interpreted from the recorded threads. The narratives intersperse explicit design contexts (for brevity only marked with number 1 – 7).

Table 7.3. Conceptual summary of task T21 with design episodes 8.4a to 8.4l

Step	**a**	**b**	**2**	**c**	**d**	**3**	**e**	**4**	**f**	**5**	**g**	**6**	**h**	**k**	**l**	**7**
1a	1.		1.			1.		1.		1.		1.				1.
1b			1.			1.		1.		1.		1.				1.
2	2.	1.		3.					3.							
3a	3.			4.					4.						2.	
3b		2.			1.				4.						2.	
4		4.		1.							1.					
5			3.			3.	4.	3.	6.	3.	3.	3.	4.	3.	3.	3.
6a		2.	2.		2.	2.	3.	2.	5.	2.	2.	2.	3.	2.		2.
6b		3.	2.	2.	2.	2.		2.	5.	2.	2.	2.	3.	2.		2.
7							1.		1.		1.		1.	1.		
8							2.		2.				2.		1.	

7.6. Visualised summary of tasks T11 and T21

I conclude chapter 7, the annotation and analysis of the empirical study by summarising the analytic results. I will visualise the two annotated problems and show them as a flow of design actions forming a narrative. In other words, the individual design contexts (DC-s) are linked into a single coherent flow of design episodes (a, b, c, etc.). In addition, I also show the sequence of the individual episodes; in this case, the coherence is achieved by chaining the individual reasoning steps (1a, 1b, 2, 3a, etc.) in between the design episodes.

The visualisation is presented in the following two figures. First, Figure 7.14 shows the action flow creating the narrative structure for task T11 – design of an active shock absorber. Next, the diagram in Figure 7.15 depicts a similar flow constructed for the task T21 – design of a paper-smoothing plant. The symbols used in the figures are straightforward and relate directly to the definitions and glossary of terms introduced in section 7.1. Thus, the three different shapes

correspond to three different levels of annotation and analysis: design contexts, design episodes and reasoning steps.

Design contexts are shown as hexagons, and each line of reasoning begins and ends with the design contexts. Each context is labelled with the same name as appears sections 7.2 to 7.5. Design episodes are symbolised by rectangles, and they are also labelled in accordance with the references used in Table 7.2 and Table 7.3. Finally, the lowest level of the RFD analysis recognised the reasoning steps, these are marked as circles in the diagrams, and are labelled with their particular referential values (i.e., 1a, 4, 6b, etc.) that are translated in the legend above.

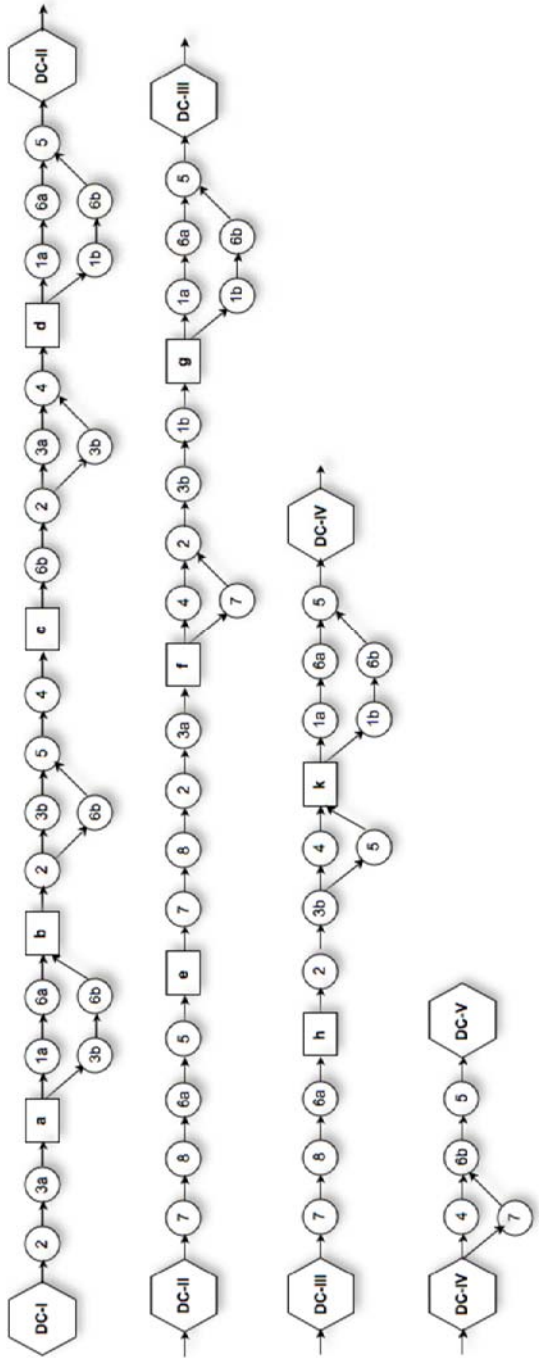


Figure 7.14. Flow of actions in the design session corresponding to problem T11 (active shock absorber)

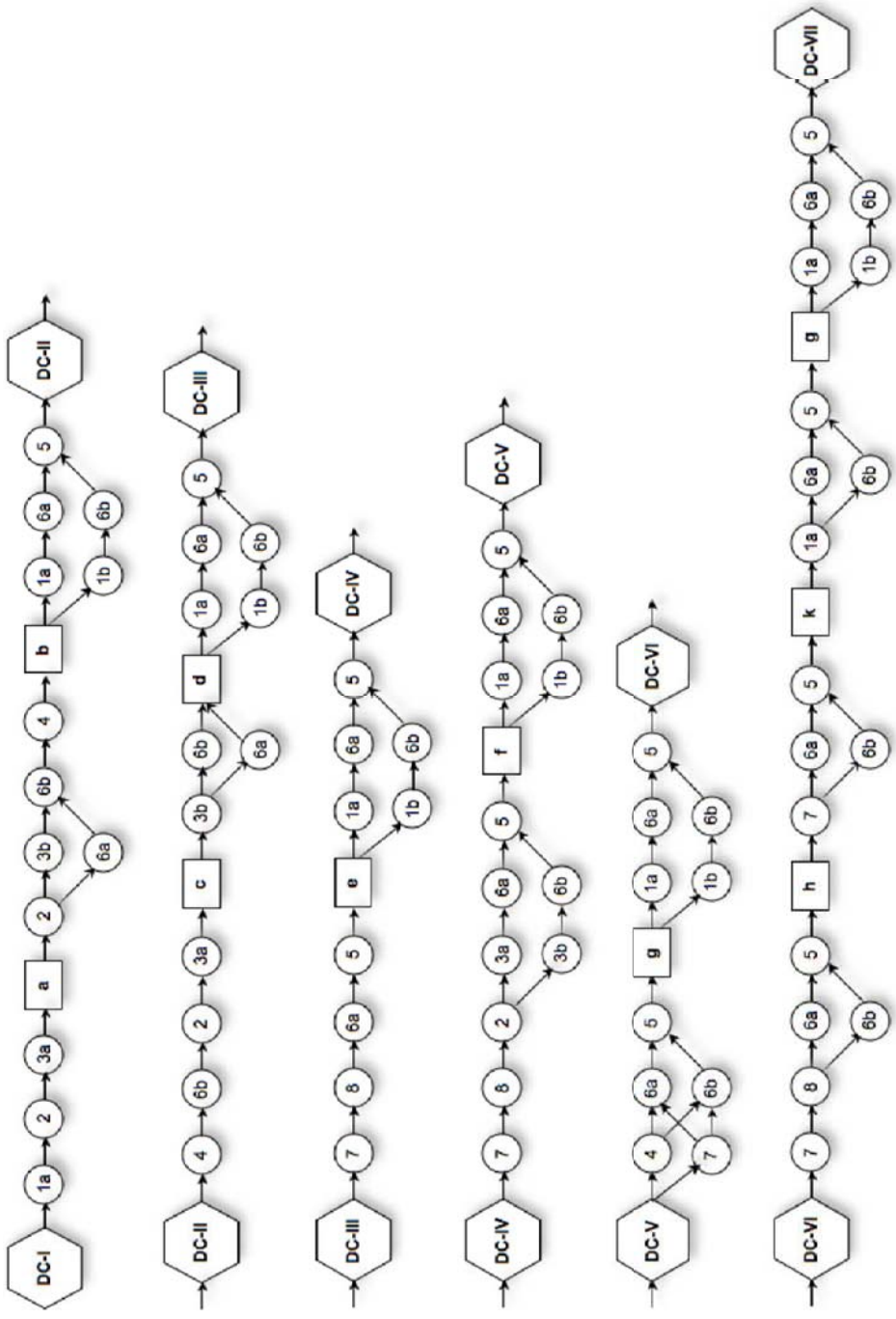


Figure 7.15. Flow of actions in the design session corresponding to problem T21 (paper-smoothing plant)

Chapter 8

Experiment Results and Implications

In addition to two experiments annotated in chapter 7, I conducted altogether 24 experiments with non-trivial design situations. All the experiments were annotated and analysed similarly as shown in chapter 7. Using the methodology and instructions for the annotation of raw data from section 7.1, my studies resulted in a set of tabular summaries for each scenario similar to those in Tables 7.2 and 7.3. The purpose of this chapter is to present these global findings and summaries.

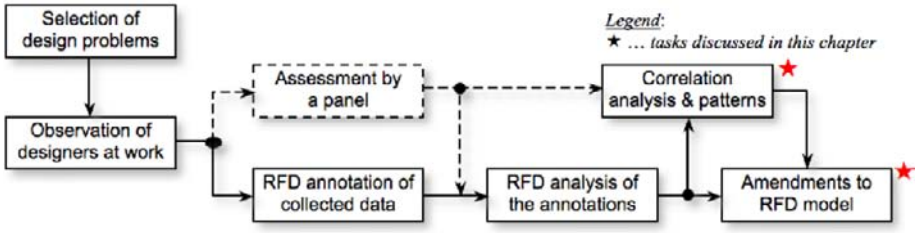


Figure 8.1. Phases of the experimental study tackled in this chapter

According to Figure 8.1, this chapter is concerned with the identification of interesting patterns across all experiments and with their analysis. However, a few improvements to the original RFD model will be identified too. Let us start with the overall summaries in section 8.1, where I identify pattern sequences re-occurring in the design process, and relate them to the individual predicates, sequences of

predicates, and the characteristics of design. The significant patterns are then super-imposed on the preliminary assessment of the design sessions presented earlier in section 6.3.

Interesting correlations are presented in section 8.2, where I also show how the innovative and routine problems perform in the conceptual terms of my RFD model. Next, I draw conclusions from these patterns and associations, and revise the original recursive model of framing in design, as there are certain implications of the patterns discussed in section 8.2 to improve and complete the model. Therefore, section 8.3 proposes a few extensions to the model from chapter 5. These are are then elaborated in more detail in chapter 9.

8.1. Summary of the experimental findings

The previous chapter concluded with the visualisation of the sequence of reasoning actions observed in the two annotated and analysed experiments. Similar visual maps were made for the remaining design scenarios. However, this form of result presentation is clumsy and difficult to generalise. The inspiration for the alternative presentation of generalised results came from history, from the statistical graph of Napoleon's march to and from Russia. It was devised by Charles Minard who graphically depicted the fate of Napoleon's army in Russia in 1816 by visualising the changing strength of the army over a period of six months. Minard's graph associated the time, strength, direction of marching, and the temperatures [Tuf01]. My take-on of the metaphor is based on the following dimensions and rules:

- The X-axis reflects the sequential order, in which design actions took place; the allowed values are discrete step numbers (e.g., 1, 2, etc.); my studies yielded a maximum of six steps;
- The Y-axis depicts the reasoning steps from the RFD analysis of each design session (e.g., 1a – articulation of a requirement, 7 – acceptability check, etc.);
- A bullet located at co-ordinates $[x, y]$ corresponds to an operation Y being observed in step X in the respective design episode (e.g., 7 – acceptability check observed as 2. step);
- A line between the bullets in the two consecutive steps corresponds to an occurrence of the particular sequence of reasoning steps (e.g., step 7 – acceptability check immediately followed by step 6a – specification refinement);

- Each occurrence of a particular sequence of reasoning steps (say, $7 \rightarrow 6$) corresponds to a line that is 0.25 of a point thick;
- Repeated occurrence of the same sequence of operations observed in the same two steps makes a particular line thicker; the increment is always 0.25 of a point per sequence;
- Thus, the thicker a line is, the more frequently a particular sequence of actions was observed and annotated.

Figure 8.2 shows the graph of reasoning sequences in its full form. As seen in the figure, the number of transitions between various design actions is very large. Altogether, there are 104 different types of transitions (links) depicted in the graph. The line intensity ranges from the thinnest 0.75 of a point to the thickest 15.25 of a point, and to make the graph more legible, I present versions that take into account various thresholds later in this section.

8.1.1. Conceptual terms and assumptions in the graph

To interpret the generalised results in Figure 8.2 I go back to the characteristics of design identified in chapter 3. These characteristics are clustered in four separate categories[1]:

1. Presence of knowledge transfer and re-use of the familiar frames;
2. Presence of the phenomenon of co-evolving design specifications and solutions;
3. Re-formulation of design problems (that is, articulation of additional requirements or constraints);
4. Reflection on partial design solutions

Knowledge transfer and familiar frames

This category covers the application of the previous experience in framing and solving the design problem, as well as various forms of reused knowledge. 'Knowledge transfer' refers to strategies such as analogy- and similarity-based reasoning, prototype and model re-use, or the application of explicit good practice. As is visible from the pattern highlighted in Figure 8.3, the "knowledge transfer" sequence occurred very frequently in the observed design experiments. Thus,

[1]The order is changed so that it is possible to present findings in a meaningful way that reflects the causality of the particular reasoning sequences.

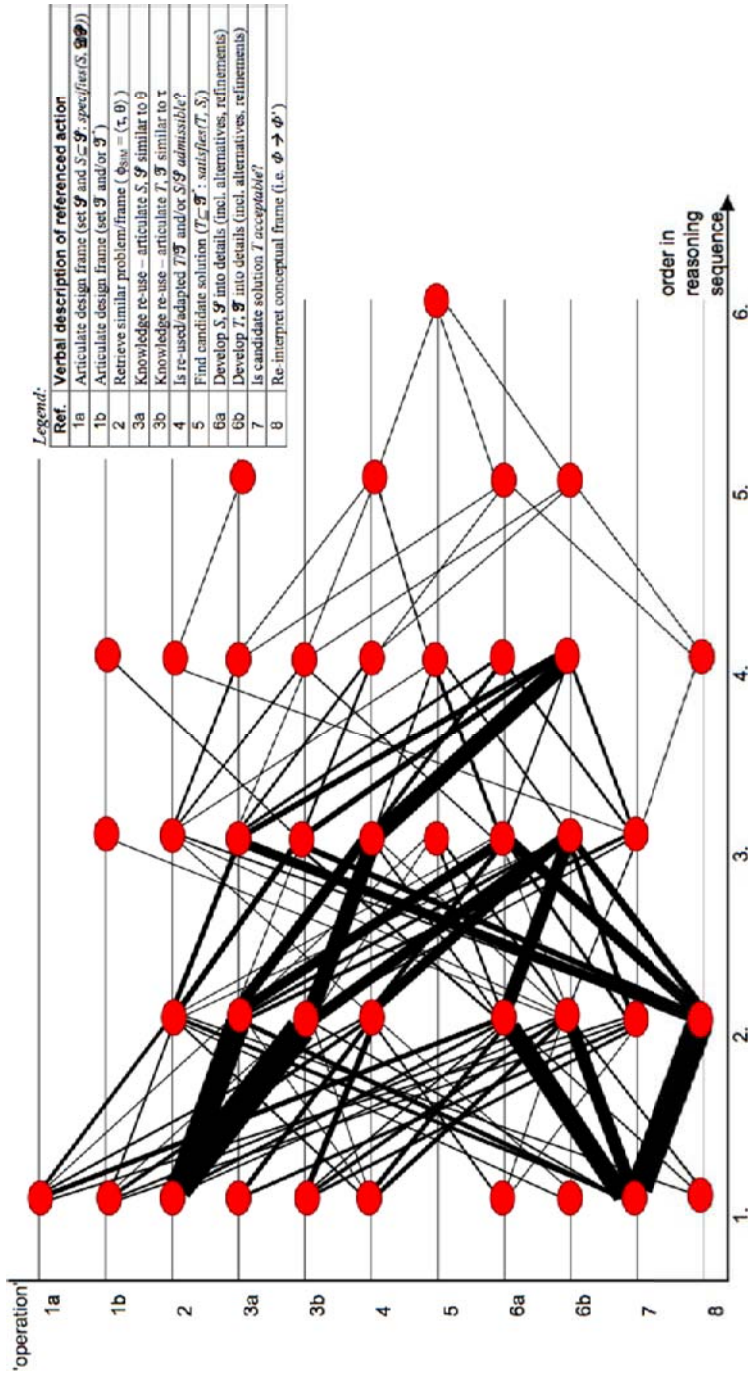


Figure 8.2. Occurrence of reasoning steps sequences aggregated over all experimental sessions (each occurrence corresponding to 0.25 thick link)

we may conclude that this assumption was upheld in the experimental work. Moreover, it is possible to recognise a typical 'forking' pattern of knowledge re-use that is repeated not only at the beginning of the design episode, but also in later steps.

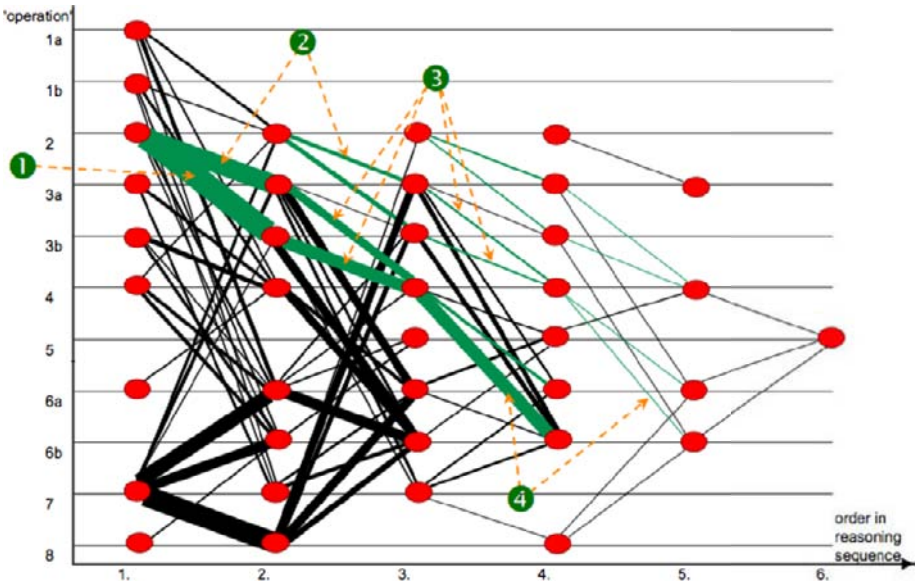


Figure 8.3. Pattern 'knowledge retrieval – re-use – application'

The "knowledge transfer" pattern consists of the following reasoning actions that typically co-occurred in my studies. The process begins with an identification of a familiar frame; that is, a past design case, a design prototype, or a model of a controller. The past frame is then used to define the conceptual objects the designer intends to use for the construction of a solution in the current problem; that is, the conceptual base $\mathscr{T}$. This transition is depicted in Figure 8.3 as a link between *action 2* (retrieval of past frame $\langle \tau, \theta \rangle$) and *action 3b* (articulation of T and $\mathscr{T}$ based on previous τ). Altogether, this transition occurred 61 times in step 1, 13 times in step 2, 4 times in step 3, and is labelled by pointer '1'.

Another transition observable as a part of knowledge transfer connects *action 2* (retrieval of past frame $\langle \tau, \theta \rangle$) and *action 3a* (articulation of S and $\mathscr{S}$ based on previous θ). This transition occurred slightly less frequently than the previous one (51 times in step 1, 11 times in step 2, and 8 times in later steps). In the figure, it is la-

belled with pointer '2'. The emerging 'fork' supports my hypothesis from chapter 4 that the past cases may serve as models for both the problem specification and the solution construction.

Typically, these actions documenting knowledge re-use in either of the two knowledge spaces forming a design frame were followed by an assessment of the relevance of a piece of retrieved knowledge to the current problem. This was typically a check of compliance with other known domain objects, partial solutions or constraints. As argued in section 5.3, such an explicit assessment can be modelled by a formal theory featuring TMS, ATMS or constraint satisfaction techniques. In the graph, it is represented by 'a merger' of two links starting in nodes $3a$ and $3b$, and ending in node 4. The occurrence of assessing the re-used specification ($3a \rightarrow 4$) is 10 times in step 1, 23 times in step 2, and 6 times in later steps. A similar breakdown for transition $3b \rightarrow 4$ is: 13 times for step 1, 34 times for step 2, and 6 time in later steps (see pointer labelled as '3').

However, the aggregated results show that after the admissibility assessment, typically one more design action occurred. Re-used components may have been refined or *adapted* (see also chapters 3 and 4 for the operation of adaptation in CBD). This adaptation is represented in Figure 8.3 by a frequent occurrence of a transition from *action 4* (admissibility check) to *actions 6a* or *6b* (refinements of sett $\mathscr{S}$ or $\mathscr{T}$). Altogether, this 'follow-up' after the explicit assessment of the transferred knowledge was observed 35 times in step 2 and 22 times in step 3 (see label '4').

Co-evolving specification and solution

From the aggregated results it is also possible to confirm a pattern corresponding to the phenomenon that was referred to in the previous chapters as a co-evolution of design specification and solution. Multiple paths in the graph can be associated with this particular feature of a design process. They are highlighted in Figure 8.4, and in general, these pattern have one thing in common, they are represented by links starting in nodes $1a/3a/6a$ (articulation of S or $\mathscr{S}$) and ending in nodes $1b/3b/6b$ (articulation of T or $\mathscr{T}$), or vice-versa.

The variability in the location of this "co-evolution" pattern is dictated by the event that triggered the re-formulation in the first place. Thus, the most frequent is the development of a solution ($T \subset \mathscr{T}^*$)

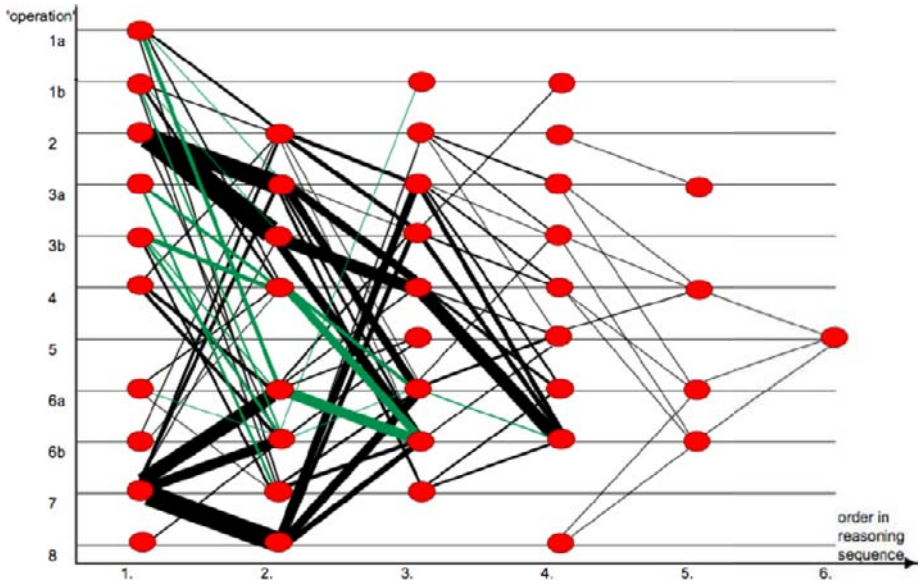


Figure 8.4. Patterns showing co-evolution of problem specification and solution

based on the similarity of the retrieved problem specification and the current problem specification ($S \subset \mathscr{S}^*$ similar to θ). This transition occurred with or without an admissibility check, and in the graph is represented by sequences $3a \rightarrow 6a/6b$ or $3a \rightarrow 4 \rightarrow 6a/6b$. A verbal account of such transitions is that after the assessment of a previously-known requirement, it was usually adapted for the current problem, and its implications were drawn in the space of design solutions.

The interaction also occurred in the reverse order; see transitions of the type $3b \rightarrow 6a$ or $3b \rightarrow 4 \rightarrow 6a$. However, judging from the frequency, this pattern is less usual than the above-mentioned one. Nevertheless, this is not a shortcoming of my theory; it supports the viewpoint that this 'solution backtalk' is often conducted at an tacit level, and features some kind of acceptability assessment of the partial solution rather than the explicit admissibility. Indeed, if we subscribe to this fundamental assumption of the RFD theory, we can note an unusually thick link between node 7 (acceptability check) and $6a$ (refinement of S or $\mathscr{S}$). This particular link is associate with the difference between a routine refinement of the problem specification and solution (that is, transitions $3a/b \rightarrow 6a/b$) and a reflective refine-

ment of the designer's understanding (i.e., transitions $7 \rightarrow 6a/b$). In other words, the "co-evolution" pattern takes place *between two* (or more) consecutive design episodes, rather than *inside a single one*.

Problem re-formulation, new requirements/concepts/etc.

Re-formulation and refinement of the designer's understanding of the design problem is also supported by the aggregated results. There are the usual ways of introducing new requirements and/or constraints to the current design situation. The term 'usual' in this context refers to knowledge re-use that is observed early in the particular design episodes and early in the design session. In the terminology of my RFD model, it is more suitable to refer to this action as the initial interpretation of requirements and constraints.

Even if we disregard the initial interpretation and commitment to the explicit knowledge about the particular design problem, the highlighted pattern in Figure 8.5 shows another strong 'talkback'. The origin of this re-interpretation of the current understanding of the design problem is the (non-)acceptability of the current solution (node 7). In many cases, the acceptability check failed to satisfy the designer's tacit expectations, and the designer explored the reasons. To that extent usually looked at the problem from a different angle; that is, transition $7 \rightarrow 8$ labelled as '1' occurred as a frequent 'first aid' to overcome the deadlock.

Nevertheless, the designer did not end his exploration with a different view. After shifting or amending his current frame, he had several choices (indicated by labels '2', '3', '4', '5', and '6' in the figure). As we can see, the most frequent was the transition resulting in the refinement of an existing statement in the explicit problem specification (node $6a$ and label '5' – 24 times). This was followed by a more radical problem re-interpretation featuring introduction of a new statement to the explicit problem specification (node $3a$ and label '3' – 21 times). An interesting link in this 'fanning' pattern is the one labelled as '2' that leads to recalling the previous knowledge (node 2). Although, this link does not prove or document the problem re-interpretation directly, it supports the position of introducing new, less common but still familiar views to resolve the tacit unacceptability of a particular sound solution.

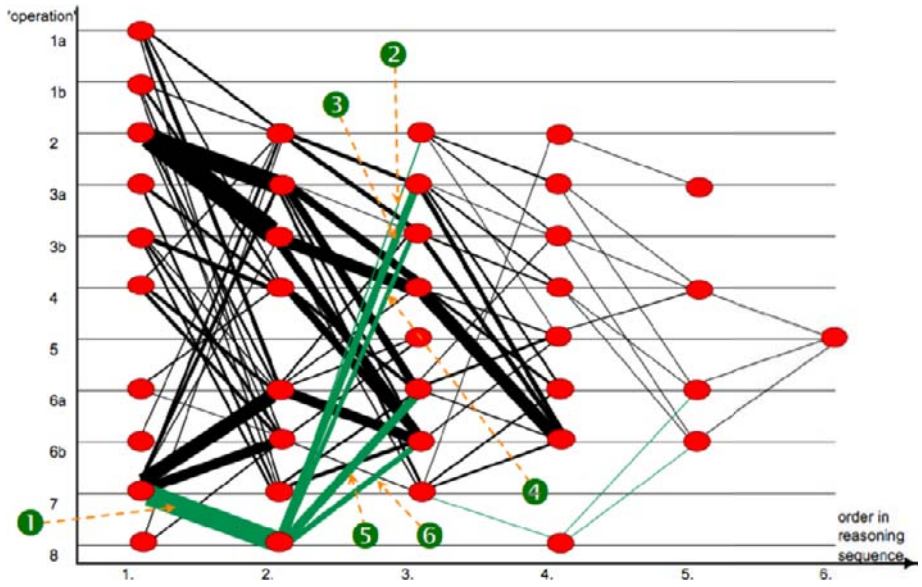


Figure 8.5. Highlighted re-interpretations of the problem understanding

Reflection on partial design solutions

Finally, let us conclude this generalisation of the aggregated results by looking at the most interesting characteristics of design – reflection on partial solutions. As mentioned earlier, the presence of 'solution backtalk' or reflective appreciation of a partial solution is typically observable between two consecutive design episodes. To provide a better picture, I present this information as a bar chart in Figure 8.6.

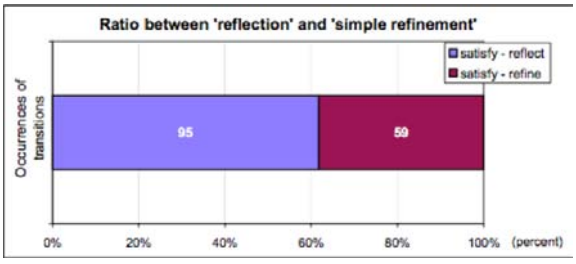


Figure 8.6. Reflection and solution backtalk vs. solution refinement

The bar graph was constructed by going through the analyses

of all design sessions and noting how many times the explicit commitment to a (partial) solution was followed by the appreciation of acceptability and the decision to extend the current perspective. The first causal relationship corresponds to sequence 5 → 7, as is depicted in Figures 7.14 and 7.15; the second one denotes the transitions 5 → 3 or 5 → 6. These *inter-episodal transitions* were not included in the aggregated Figure 8.2. The reason is that Figure 8.2 depicts what happened *inside* design episodes, whereas the reflective 'solution backtalk' (Figure 8.6) occurred between the episodes. Moreover, I did not look at any arbitrary episode pairs but focused on what happened after the explicit commitment to a particular context (in chapter 7, these explicit commitments were annotated as 'DC' entities, say, T11/DC-II or T21/DC-VI).

Figure 8.6 shows that it was possible to distinguish between the episodes in which the proposed partial solution was only elaborated into details and the episodes featuring inarticulate decisions. I believe it is appropriate to talk about reflection as a knowledge-level representation of the acceptability assessment by a designer. Thus, transitions corresponding to the '*satisfy* → *reflect*' portion from Figure 8.6 are depicted in Figure 8.2 as paths starting from node 7 (acceptability check). Any other refinement or elaboration of a partial solution is labelled as '*satisfy* → *refine/extend*' in Figure 8.6, and in Figure 8.2, the source of such refinements are nodes 1*b*, 3*b* or 6*b*.

Nevertheless, the bar graph depicting the ratio of inarticulate, tacit exploration versus explicit and straightforward refinements supports my hypothesis that design is a reflective process. Following possible paths after the check of tacit acceptability, the process of design framing is also reflective. Above all, regardless of whether the solution was extended, refined, or re-interpreted in a new frame, the bar graph in Figure 8.6 supports the use of the adjective recursive in my Recursive model of Framing in Design. The observed design sessions exhibited between four and ten such recursive iterations from a partial solution towards a more complete and better-understood one.

8.1.2. RFD model supported by the aggregated graph

After presenting the aggregate results at a generic level, let us look at how they support the RFD model from chapter 5. First, the essential tenets of the recursive model of framing in design are shown next to

the interpreted aggregated results. Then, the individual steps and levels defined for the RFD model are analysed step-by-step.

The original recursive model of framing and re-framing in design was presented as a sequence of three interacting knowledge-level actions, namely: '*specifies*', '*satisfies*', and '*acceptable*'. The mutual interaction between these predicates was achieved by an interchange of data and control during the design session. An auxiliary predicate '*modifyFrame*' served as an abbreviation of the recursive iteration defined in Eq. 8.1. Let us now associate the aggregated results from Figure 8.2 with the RFD model using the individual levels as the structure for our analysis.

$$modifyFrame(\Phi) \iff \exists \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle : S' \subset \mathscr{S}^*_N \\ \wedge specifies_{\Phi_{NEW}}(S', \mathscr{D}\mathscr{P}) \tag{8.1}$$

Level 0. $\boldsymbol{specifies_{\Phi}(S, \mathscr{D}\mathscr{P})}$

Level 0 denoted the articulation of an initial conceptual frame for the design problem $\mathscr{D}\mathscr{P}$. The construction of an initial frame started with a reference to the familiar vocabulary and continued with an attempt to describe the current problem $\mathscr{D}\mathscr{P}$ in such vocabulary. The result of this initial framing was an explicit articulation and commitment to a pair of two conceptual sets $\mathscr{T}$ and $\mathscr{S}$. Set $\mathscr{T}$ contained the basic conceptual terms the designer decided would be available for solving the design problem $\mathscr{D}\mathscr{P}$. Set $\mathscr{S}$ contained the same for problem specifications.

The initial framing of the design problem $\mathscr{D}\mathscr{P}$ in the familiar terms was indeed observed during the experimental sessions. Each design session (for example, T21 in section 7.4) started with a selection of terms from the design brief that expressed the requirements; in the terminology of my model, this was action 1*a* (articulation of explicit set $S \subset \mathscr{S}^*$). The alternative step referred to a familiar frame, from which the designer extracted further details to extend the design brief, to translate it into engineering language, and to define main concepts for a solution construction. In other words, the sessions could also start with the *action 2* (knowledge retrieval), and typically continued with an adaptation of the previous specification or conceptualisation (3*a* and 3*b*).

Thus, level 0 is confirmed in the aggregate results, and since the "knowledge transfer" pattern was discussed in section 8.1.1, it will not be repeated again; Figure 8.3 applies to this section as well.

Level 1. $\boldsymbol{satisfies_{\Phi}(T, S)}$

According to the definition of predicate '*satisfies*', the purpose of this reasoning step was to find an admissible solution. A solution was admissible in respect to a particular explicit problem specification S that was decided at evel 0 as sufficiently specifying problem $\mathscr{DP}$. A complete or, more often, a partial solution T was sought in the applicable problem solving theory $\mathscr{T}^*$. We did not delve into the methods how this solution construction happened in the particular sessions, we were interested in the explicit commitments to something that could be viewed as a candidate solution. In the annotated sessions, the occurrence of such milestones was labelled as an explicit achievement with a particular problem specification S, which initially contained between two and four key requirements. In many cases, this initial specification was accompanied by an abstract drawing depicting the intended approach to the problem, that is, a simple solution model T. The second explicit milestone contained a solution model for all design sessions. For instance, session T21 in section 7.4 produced the first abstract model in explicit context T21/DC-II as a result of choosing admissible concepts during the preceding episodes T21.a and T21.b.

Taking into account the aggregate results, the pattern in Figure 8.7 labelled as '1' shows a representative sequence that can be associated with predicate $satisfies(T, S)$ in the early phases of design. Since this predicate may also occur in later phases, the alternative patterns are also shown in Figure 8.7 with labels '2', '3' and '4'. Nevertheless, the aggregate results seem to confirm the sequential order and the position of predicate '*satisfies*' in respect to '*specifies* or '*modifyFrame*' (this will be discussed further down).

Level 2. $\boldsymbol{acceptable_{\mathscr{DP}}(T)|\Phi}$

The next level according to the definition of RFD model deals with the appreciation of the solution tentatively proposed as a result of the '*satisfies*' phase. We know that a designer works within a particular conceptual frame and may try several alternative solutions, each of

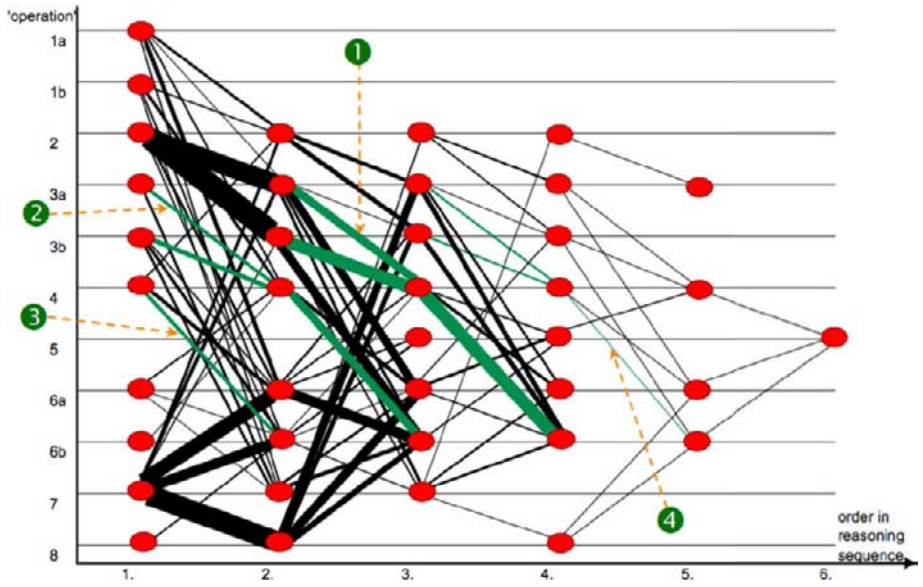


Figure 8.7. Pattern for a construction of initial admissible solution

them being explicitly sound and admissible in respect to the explicit constraints. However, once the commitment is made to a candidate solution, its compliance with the inarticulate expectations needs to be assessed. This alignment of the explicit conceptual frame and the tacit expectations was defined as a solution acceptability check, whereby the designer assesses solution T against the actual problem $\mathscr{D}\mathscr{P}$, rather than against its explicit specification S.

As noted in section 8.1.1, the acceptability check was an important decision the designer was making regularly during any design session. For instance, session T21 (section 7.4) is a typical representative with five occurrences of acceptability assessment. In the aggregated graph, the acceptability check is associated with the paths starting from node 7; a typical sequence is $7 \rightarrow 8 \rightarrow 6a$ (see the highlighted patterns in Figure 8.8). Figure 8.8 on one hand confirms the correct location of the solution acceptability assessment in the design process. However, the next assertion by the RFD model argues that if the solution acceptability check fails, it points to an incomplete design frame Φ. The RFD model suggested that a remedy for an unacceptable solution is the articulation of a new frame Φ_{NEW}. In the aggregated results (Figure 8.8), this causal relationship between

the acceptability check (node 7) and a reference to new conceptual terms of an amended design frame (node 8) is visible in the shape of a thick transition labelled as '1' and highlighted in **light blue**.

After the amendment of a conceptual frame, the designer makes modifications or refinements in the explicit problem specification (sets S and $\mathscr{S}$) or in the problem solving theories and models ($\mathscr{T}$ and T). We leave these amendments for the discussion at the next level; however, one finding must be noted that is not entirely accounted for by the RFD model. Namely, the pattern of transitions starting in node $7 \rightarrow 6a$ or $7 \rightarrow 3a$ (refinement or extension of an explicit problem specification), and in parallel link $7 \rightarrow 6b$ (solution refinement). See labels '2' and '3' in Figure 8.8.

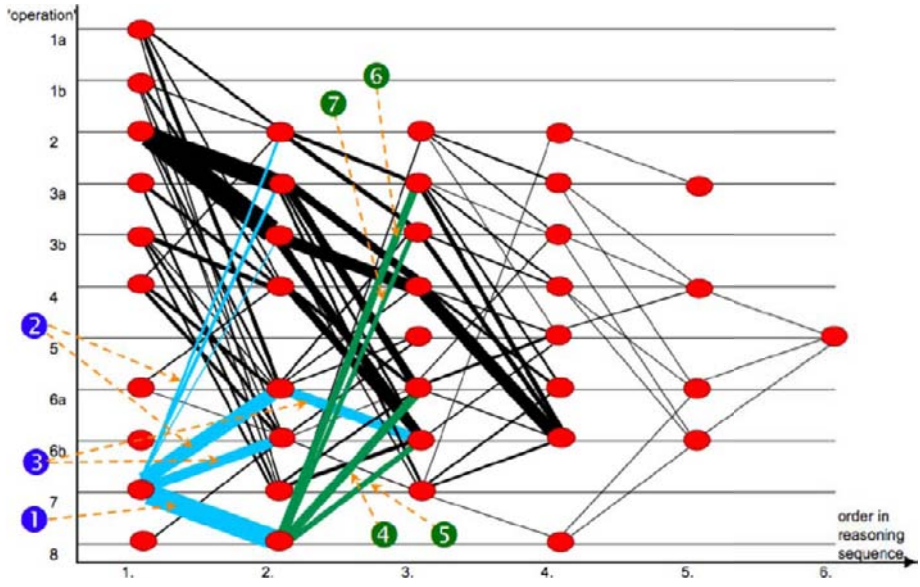


Figure 8.8. Patterns from design episodes showing acceptability check

These transitions were observed frequently, and they differ from the predictions made by the RFD model in the fact that no particular frame amendment accompanied these extensions and refinements. Thus, it seems that there are several options for the designer to choose from after assessing the acceptability of a solution. First, he may commit to a new frame; alternatively, he may refine the problem specification in the existing frame. We return to this particular pattern in sections 8.2 and 8.3, where the RFD model is re-visited and

improved. However, the essence of the original model, the presence of a tacit acceptability check was confirmed by the experiments, too.

Level 3. $\boldsymbol{modifyFrame(\Phi)}$

As mentioned earlier, this knowledge-level action follows the acceptability assessment described in the previous paragraphs. We learned that there were several paths from node 8 in the aggregated graph; see the highlighted links in Figure 8.8. Some of these follow-up actions aimed to refine the designer's understanding of the problem (for example, label '4' pointing to sequence $8 \rightarrow 6a$ and label '5' pointing to sequence $8 \rightarrow 6b$). Alternatively, the designer could commit himself to more significant extension of the explicit problem specification or solution (sequence $8 \rightarrow 3a$ labelled as '6' or sequence $8 \rightarrow 3b$ labelled as '7', respectively). All these different refinements and extensions originated in the modified design frame; that is, in the different perspective the designer applied to the investigated design problem $\mathscr{DP}$.

This observation supports the correctness of the RFD model in respect to the re-framing operation. Nevertheless, my definition of re-framing repeated in Eq. 8.1 does not account for the observed variability. In principle, the predicate defining the operation of re-framing is correct but it needs to be refined in response to the empirical observations. This newly emerged pattern is already incorporated in the RFD model; only it is defined incompletely. Therefore, in sections 8.2 and 8.3, and later in chapter 9 we make a clearer distinction between the refinement and extension of the design frames.

Recursion and iteration (level 3. → level 1.)

The amendment of a design frame is not conducted self-purposely; its objective is to bring something new into the explicit knowledge about the design problem. This 'something new' is modelled by an exploratory probe whose aim is to restore or achieve the acceptability of a design solution. Therefore, the designer has to validate the tentative proposal following the frame amendment, and the typical way to do it is to construct a new (partial) solution.

The recursion and iteration is observable in the fact that each design session ran in more than one design context (for example, DC-II to DC-VII in section 7.4). In the analysed records, iteration is also visible as a repeated sequence of design episodes leading from the old

to the new explicit achievements (see Figure 7.14 and Figure 7.15 in section 7.6). Nevertheless, the recursion does not always require the amendment of a design frame! This is another inconsistency with the RFD model, which can be resolved by an improved definition of predicate '*modifyFrame*'. However, before defining the extensions to the RFD model formally, let me show the patterns discussed above in a broader context.

8.2. Correlations between patterns and types of problems

Let us now look at the correlations of the patterns observed in the aggregated results and the types of design problems, and we we start by showing how the ratio between the reflective development and the explicit (simple) refinement of design understanding changes for the different types of design problems. The ratio reflects the occurrences of transitions 5 → 7 → '*any*' (that is, satisfaction followed by an acceptability assessment) compared with transitions 5 → 3/6 (that is, satisfy partially and refine without any acceptability checking).

Another interesting distribution was identified at levels 2 and 3 in the previous section; thus we look at how frequently each of the following transitions occurred in a particular class of problems:

- 4 → 3/6 (admissibility check followed by an extension/refinement),
- 7 → 3/6 (acceptability check followed by an extension/refinement), and finally
- 7 → 8 → 3/6 (problem extension/refinement following in the amended frame)

In order to give structure to these comparative analyses, the experimental design sessions were categorised into complementary groups. The categorisation is the result of the auxiliary assessment provided by a panel that was elicited before my analysis summarised in section 8.1. These findings also support the proposals for extending the RFD model that were hinted at in the previous section. The categories selected for the comparative analysis include the following:

1. Overall distributions (all experimental sessions);
2. Distributions for innovative and routine sessions;
3. Distributions for superficial solutions and more elaborated, complex ones;
4. Distributions for conceptually detailed and abstract solutions

Note that the classification of the individual sessions into particular categories was done for each design session based on the explicit records and drawings from that session. In other words, in practice it is not true that, for instance, all designs of shock absorbers must always be innovative. Only the design session I observed was found innovative or routine, simpler or more complex by the panel, based solely on the captured records and observed behaviours.

We start with the ratios between the reasoning sequences for all design problems that were annotated and analysed. The distribution showing the ratio between the occurrences of reflective development and simpler, explicit problem refinement is shown in Figure 8.9. The numbers in data labels correspond to the absolute occurrences of respective reasoning sequences. In relative terms, the ratio was $reflective : explicit = 61.7\% : 38.3\%$.

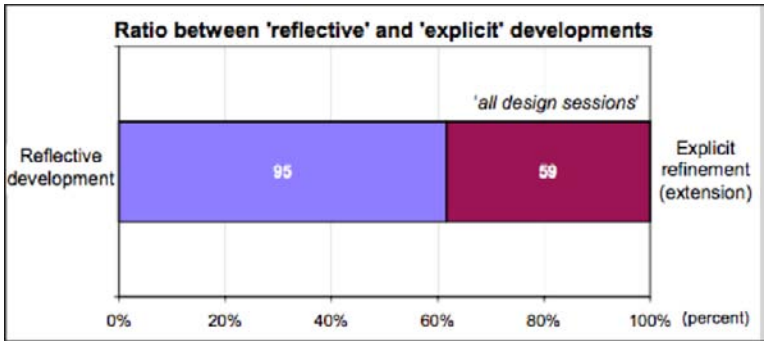


Figure 8.9. Ratio 'reflective development' vs. 'explicit' refinement of design

The bar chart in Figure 8.10 shows a similar distribution for the operations that followed the assessment of an explicit admissibility and tacit acceptability. The ratios and percentages are later compared with those for the particular classes of problems to confirm the expectations from chapters 3 and 4. The respective overall ratios are as follows:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = 65.4% : 34.6%;
- Design refinement/extension vs. re-framing following acceptability check = 55.5% : 44.5%;
- Tacit refinement vs. tacit extension of design = 83.7% : 16.3%;

- Re-frame & refine vs. re-frame & extend = 59.4% : 40.6%

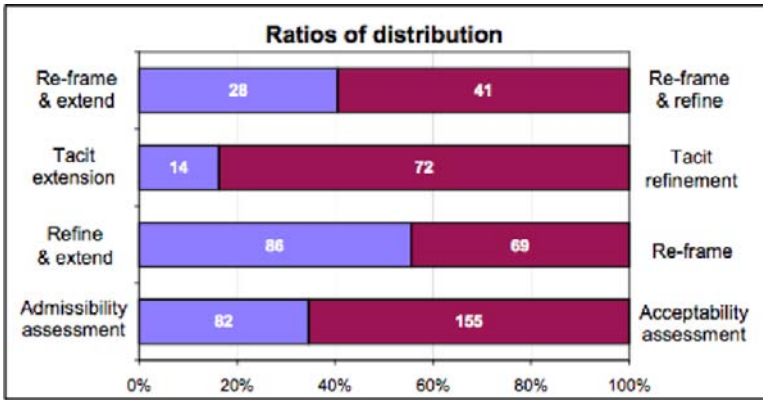


Figure 8.10. Ratios between the consequences following different assessments

8.2.1. Distributions between innovative and routine tasks

The classification of the individual design sessions into routine and innovative ones is based upon the overall impression provided by the panel, which was presented in section 6.3 (Table 6.3). Most sessions exhibited features that were considered as 'innovative' (16 tasks) compared to 11 tasks showing 'routine' features. Figure 8.11 shows the distributions between the 'explicit' and 'reflective' problem development for the two classes. The classes are characterised using percentage ratios rather than absolute frequencies. As can be seen, it is indeed the case that innovative problems are correlated with reflective reasoning, whereas routine problems are correlated with the explication of implicit features.

- Innovative problems: $reflective$: $explicit$ = 69.5% : 30.5% (data series at the bottom);
- Routine problems: $reflective$: $explicit$ = 46.7% : 53.3% (data series at the top)

Figures 8.12 and 8.13 show the distributions of the reasoning sequences for the innovative and routine problems, respectively. As could be expected, innovative problems rely on re-framing and tacit reasoning more often that the routine ones:

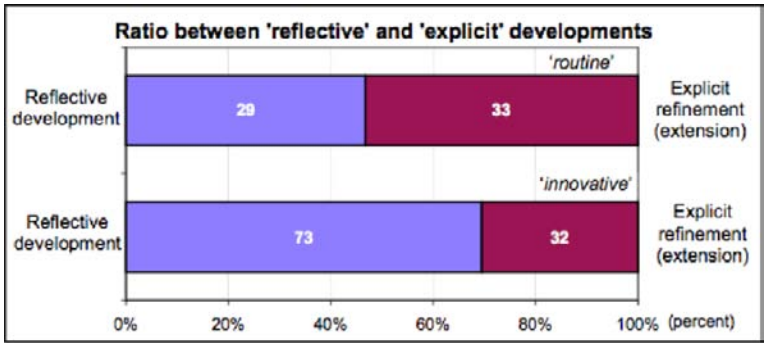


Figure 8.11. Ratio 'reflective development' vs. 'explicit' refinement/extension

Innovative problems:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = 69.5% : 30.5%;
- Design refinement/extension vs. re-framing following acceptability check = 48.8% : 51.2%;
- Tacit refinement vs. tacit extension of design = 84.7% : 15.3%;
- Re-frame & refine vs. re-frame & extend = 40.3% : 59.7%

Routine problems:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = 46.8% : 53.2%;
- Design refinement/extension vs. re-framing following acceptability check = 70.2% : 29.8%;
- Tacit refinement vs. tacit extension of design = 81.8% : 18.2%;
- Re-frame & refine vs. re-frame & extend = 50.0% : 50.0%

The first significant difference between the two classes is in the ratio between the 'explicit' and 'reflective' problem developments. The innovative problems have this ratio significantly in favour of reflection, whereas with routine problems, the ratio is in favour of explicit extensions of problem specification (see Figure 8.11). The innovative problems also show higher percentage of reflective sequences (69.5%) than the routine problems (46.7%). This is precisely reversed for the explicit extensions and refinements – 30.5% for the innovative class and 53.3% for the routine class.

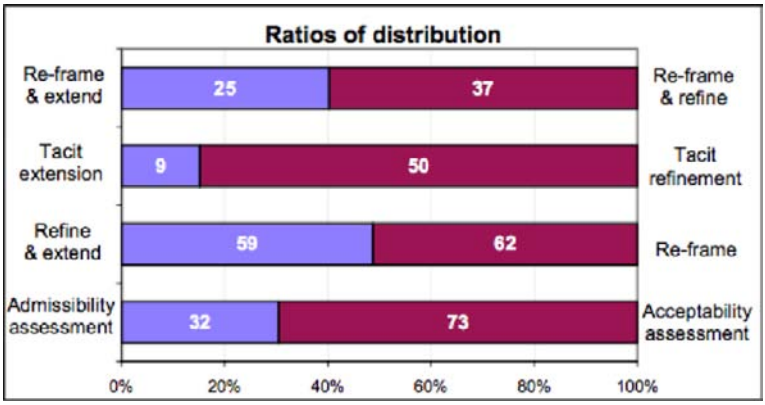


Figure 8.12. Ratios between consequences of tacit vs. explicit assessments (innovative problems)

A similar pattern is observed in the distribution between the methods for assessing partial solutions (Figures 8.12 and 8.13). For innovative problems, the reflective and inarticulate assessment of acceptability (69.5%) is more typical than the assessment of explicit admissibility (30.5%). On the contrary, routine tasks feature a more balanced assessment, and they slightly tend to prefer the explicit checks of admissibility (53.2%) to the acceptability (46.8%). Following from these ratios, the next feature is not surprising: More innovative problems show almost even distribution between problem refining and reframing (48.8% and 51.2%, respectively). The same distribution for the routine class is shifted significantly towards refinements (70.2%), giving less space to problem re-framing (29.8%).

The prevalence of reflective reasoning for the innovative tasks and the lack of problem re-framing for the routine ones uphold the characteristics associated with the respective categories. Innovative problems tend to be vaguer and under-specified, thus requiring more reflection, and the amendments to problem understanding are more radical, including re-framing. On the contrary, routine tasks are better defined; they need clarifications and refinements rather than significant perspective shifts. Routine tasks are not ill-structured problems, therefore, explicit criteria for the solution admissibility are typically defined in advance. With a large number of explicit criteria available, there seems to be less need for tacit assessments. Vaguely defined or

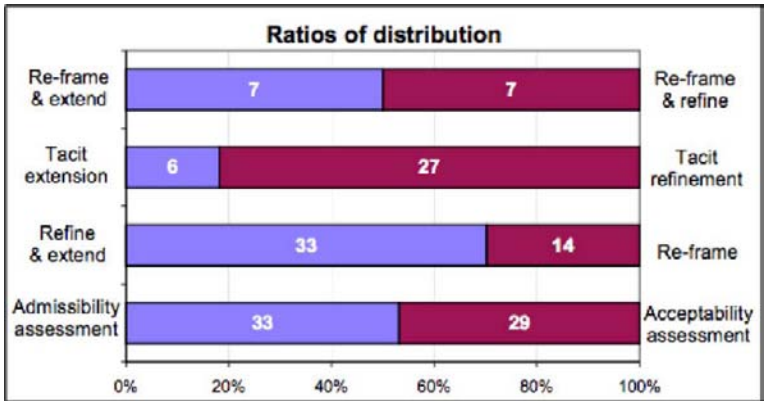


Figure 8.13. Ratios between consequences of tacit vs. explicit assessments (routine problems)

missing criteria of admissibility in the 'innovative' problems seem to force the designer to rely on the tacit assessments of acceptability.

In addition to these ratios, the innovative problems show scores that are well above the average scores calculated for all experimental sessions. Subsequently, routine problems are below the average scores presented in Figures 8.9 and 8.10, which complies with the perception that innovative solutions usually outperform the 'average' ones.

8.2.2. Distributions for superficial and elaborated solutions

The classification of the design sessions in respect to producing superficial solutions (often only control algorithms) and more elaborated ones is based upon Table 6.2 in section 6.3.2. The number of sessions in the two categories is almost evenly distributed – 14 superficial and 13 elaborated solutions. Figure 8.14 shows distributions between explicit and reflective problem development in the two classes:

- Elaborated design solutions: $reflective : explicit = 68.5\% : 31.5\%$;
- Superficial design solutions: $reflective : explicit = 55.0\% : 45.0\%$

As seen in Figure 8.14, sessions producing more elaborated solutions tended to rely more on reflective reasoning (68.5%) rather than direct refinements to the problem specification (31.5%). On

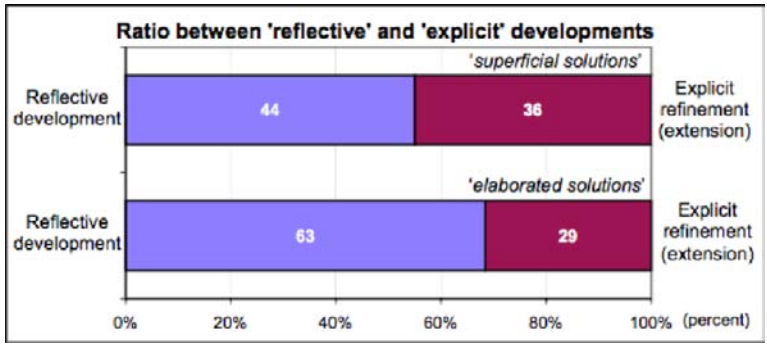


Figure 8.14. Ratio 'reflective development' vs. 'explicit refinement' of design

the other hand, the gap between reflective development and explicit refinements/extensions was much less significant for simpler, more superficial solutions – 55% reflective and 45% explicit. In comparison with the previous section, these are remarkably similar ratios. Only in this case, more elaborated and complete solutions tended to force the designer to reflect more frequently (68.5%) than the straightforward, superficial solutions (55%). However, the gap is narrower than it was between routine and innovative tasks, which suggests that innovation does not necessarily come hand in hand with solution complexity!

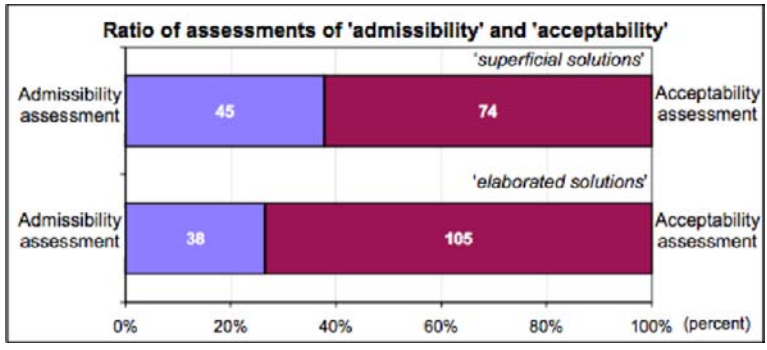


Figure 8.15. Ratio 'admissibility assessments' vs. 'acceptability assessments' of partial solutions

From Figure 8.15 it is possible to conclude that more elaborated solutions may have required more tacit assessments of the acceptability than simpler, superficial solutions. Solutions that were more

complex also contained more inarticulate assessments of acceptability (73.4%) than the explicit checks of admissibility (26.6%). The ratio for the superficial solutions shows a similar trend; the acceptability checks were more frequent (62.2%) than the checks of explicit admissibility (37.8%).

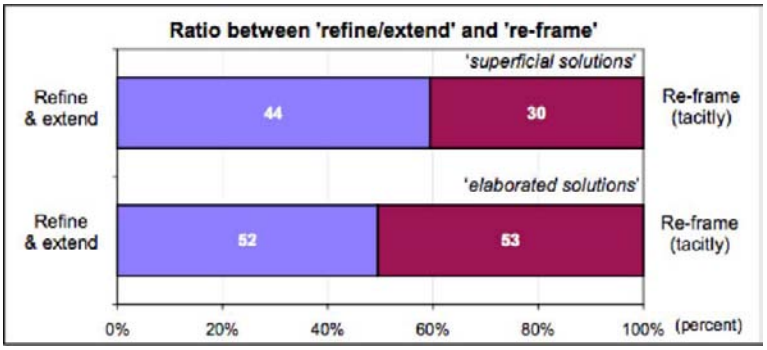


Figure 8.16. Ratio between 'solution refinements' and 're-framing'

Correlation findings from the previous figures are confirmed in Figure 8.16. The elaborated solutions seemed to require more effort from the designer in understanding the particular details he wanted to elaborate. A re-framing sequence in reasoning occurred in 50.5% cases, compared to only 40.5% for the superficial design solutions. The superficial solutions tended to exhibit more refinements to the specification (59.5%) than re-framing (40.5%). This suggests that the designer needed to introduce fewer conceptual terms and requirements for superficially handled problems than for those that yielded more elaborated, complete solutions.

8.2.3. Distributions for detailed and abstract solutions

A similar pattern between reflective development of a design problem and the direct, explicit refinements to problem specification is also observable when comparing conceptually detailed and conceptually abstract solutions. Conceptually detailed solutions included details of the implementation, whereas conceptually abstract ones usually contained only block diagrams and strategies rather than implementations. Figure 8.17 shows the distributions between explicit and reflective problem development in the two classes:

- Conceptually abstract design solutions: $reflective : explicit = 63.1\% : 36.9\%$;
- Conceptually detailed design solutions: $reflective : explicit = 57.0\% : 43.0\%$

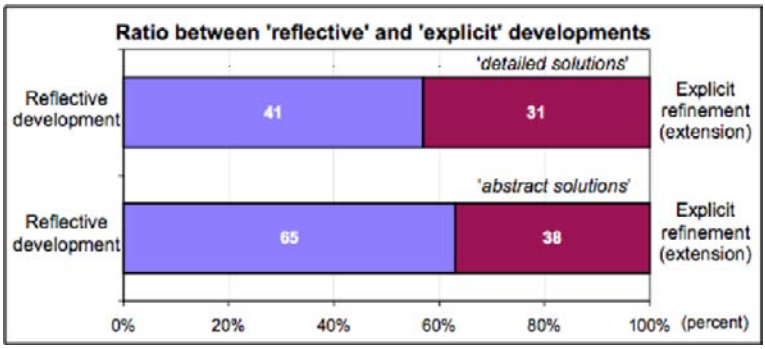


Figure 8.17. Ratio 'reflective development' vs. 'explicit refinement' of design

As one can see from Figure 8.17, there is no significant correlation between the level of conceptual detail and the frequency of reflection. Conceptually abstract solutions tend to provoke more reflection (63.1%) than direct refinement to problem specification (36.9%), but the same holds for the conceptually detailed solutions – reflection observed in 57% cases, direct refinement in 43%. There is a marginal difference in the ratios, which goes slightly in favour of conceptually detailed solutions. This is understandable – if the solutions are conceptually abstract, it may be difficult to refine them without any new frames, exploration or probing.

Nonetheless, the conclusion from this particular visual representation is that there is little correlation between the conceptual abstraction and tacit, reflective reasoning strategies. In some cases, it was possible to produce an abstract solution from a re-used template – directly refining or extending it. In other sessions, even abstract block diagrams may have required more effort.

8.3. Revision of the RFD model

As shown in this chapter, the empirical results and the aggregated graphs confirmed the correctness of the RFD model. Nonetheless,

a few patterns emerged in the analysis that are not sufficiently accounted for by the model defined in chapter 5. This is not a major shortcoming of my theory and the model; it rather supports the fact that this book is concerned with ill-structured design problems, and that the formalisation is iterative by its nature.

In the next chapter, I look at these refinements in more detail. However, let me conclude this chapter by presenting the implications of the results, with a brief enumeration of the patterns that need further elaboration:

1. Designers seem to resolve their acceptability assessments either by a direct refinement of the current design frame,or they articulate a new design frame.
2. Acceptability assessment need not always lead to a frame amendment; there is also an operation of frame refinement.
3. The actual frame amendment (problem re-framing) exhibits two distinct types. First, it is re-framing as a result of an admissible solution not being accepted; let me refer to this type as problem specification refinement via re-framing.
4. Second, there is re-framing 'proper' or a conceptual re-framing that is triggered by the non-existence of an admissible problem solving theory; that is, there are contradictions in the theory that cannot be overcome by refining the problem specification in one frame.

The patterns mentioned above require definitions of additional predicates that would enable recursion both with and without frame amendment. However, as we shall see in chapter 9, the extended RFD model is only a more complete version of the original RFD model from chapter 5.

This page intentionally left blank

Chapter 9

Extensions to the RFD Model

As shown in chapter 8, the Recursive Model of Framing in Design is a reasonable approximation of complex reasoning in design. Nevertheless, the empirical findings also showed that the model introduced in chapter 5 was not complete and could be further improved. A major extension is needed to account for the situations when the explicit problem specification and solution may be refined irrespective of shifting the design frame. In chapter 8 these situations were represented by sequences $7 \rightarrow 3/6$, and were distinguished from simpler, explicit refinements that can be produced, for example, by deduction. I will refer to the specific form of reasoning that aims to refine the current conceptual frame rather than replace it as *a design frame refinement*. Its chief attribute is that all the improvements occur without any frame amendment.

Another pattern that emerged from chapter 8 showed that there were at least two *different types of re-framing*. One type of frame amendments aimed to refine the designer's commitment to a particular problem specification or solution by borrowing the vocabulary of different conceptual frames. The current conceptual frame could be *extended* in terms of new explicit problem specification; that is, new statements could be added into it to specify a new objective that could not be originally expressed. This kind of re-framing is less complex than the one defined in chapter 5. In the aggregated graphs it corresponds to sequences $7 \rightarrow 8 \rightarrow 3/6a$.

Also my original definition of re-framing has to be revised and presented as a specific form of frame amendment. To distinguish it from the frame extension mentioned earlier, in this case the *design*

frame is amended conceptually. In other words, an entirely new conceptual apparatus is needed to construct a meaningful design frame. Unlike in the previous case, this frame shift focuses on the conceptual primitives, terms and objects for the solution construction. Since the objects are changed in this operation, the applicable domain theories change as well.

Let me define the new patterns similarly as we defined the original RFD model. Definitions of terms and core relations are not repeated, because these were upheld in the experiments and remain unchanged. Section 9.1 defines additional predicates to implement the above-mentioned improvements. Section 9.2 presents examples of the newly defined reasoning schemas. For a summary of the extended model, see also [DZ02].

9.1. Recursive model of framing in design (2.)

Extensions in this section supersede the claims made in section 5.2. The building blocks of the extended RFD model remain the same as in chapter 5:

1. a problem solving theory $\mathscr{T}^*$,
2. an explicit problem specification S,
3. a problem solving model T, and
4. a conceptual design frame Φ

The extended RFD model is defined as a sequence of interactions between two predicates '$satisfies$' and '$specifies$', as introduced in chapter 5. predicate '$specifies$' explicitly articulates the problem specification, and '$satisfies$' attempts to solve the explicitly specified problem. These two actions have a potential to generate and change the explicit design knowledge. They are complemented by the third predicate '*acceptable*' that acts as a switch deciding on whether the design process continues or it may be successfully concluded by an articulation of a design solution.

As in chapter 5, several auxiliary predicates are needed to express the new patterns, so that the model can be read more intuitively as a sequence of decisions followed by actions. According to the experimental findings, the simplest modification to the interpretation of the design problem is the explication of a statement that is believed to refine the current problem specification. If such a statement can be

articulated using design frame Φ and the vocabulary of the currently chosen conceptualisation $\mathscr{T}$ one continues the flow in Figure 9.1 at level 3. The predicate for frame refinement is defined by Eq. 9.1.

$$refineSpec_{\Phi}(S) \iff \exists s \in \mathscr{S}^*, S \subset \mathscr{S}^* : \\ S' = S \cup s \wedge specifies_{\Phi}(S', \mathscr{D}\mathscr{P}) \tag{9.1}$$

This predicate does not modify the actual design frame. It only moves the designer's attention to the statement that already existed in the problem interpretation; now an explicit commitment to that statement was amde. The fact that the design frame remains untouched is emphasised by treating Φ as a parameter circumscribing the space for the possible refinements. The predicate in Eq. 9.1 corresponds to transitions $reflect \rightarrow refine$ spotted in the experiments.

Another auxiliary definition expresses a more complex decision that attempts to refine the explicit problem specification using a new frame. Thus, there is still a refinement of the current problem specification; however, new statements are introduced from a new design frame rather than the existing one. The difference from the predicate in Eq. 9.1 is that this is a sequence of $reflect \rightarrow reframe \rightarrow extend$ from the experiments. These more complex extensions cannot be done without changing the conceptual frame; that is, with the currently used conceptual categories from set $\mathscr{S}$. Eq. 9.2 contains a formal definition of the decision to resolve the tacit non-acceptance by committing to the terms borrowed from a new design frame (new view on the problem).

$$reinterpret(S, \Phi) \iff \exists \Phi_{NEW} = \langle \mathscr{T}, \mathscr{S}_N \rangle : \\ S' \subset \mathscr{S}_N^* \wedge specifies_{\Phi_{NEW}}(S', \mathscr{D}\mathscr{P}) \tag{9.2}$$

And finally, a decision that can be most disrupting to the current design frame and the interpretation of the problem is defined in Eq. 9.3. What is changed in this particular situation, are the conceptual means, primitive concepts of the current design frame for solving the problem. The problem conceptualisation $\mathscr{T}$ that formed the basis of the old frame Φ is given up and replaced with $\mathscr{T}_N$. A new frame is articulated so that the explicit specification of the problem (that is, $S \subset \mathscr{S}^*$) is made into a consistent and admissible specification of

the actual design problem $\mathscr{DP}$ – albeit in a modified design frame, modified context, and with a modified conceptual vocabulary.

$$reframe(\Phi) \iff \exists \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle : T' \subset \mathscr{T}_N^* \wedge \\ \wedge S'' \subset \mathscr{S}_N^* \wedge specifies_{\Phi_{NEW}}(S'', \mathscr{DP}) \tag{9.3}$$

Note an interesting feature in the definition re-framing in Eq. 9.3. Unlike any previous definitions, this one is amending set $\mathscr{T}$ (problem conceptualisation). The newly articulated conceptual foundation $\mathscr{T}_N$ inevitably leads to re-considering the choice of a generic domain theory DT, and eventually to the definition of a new problem solving theory $\mathscr{T}_N{}^*$. However, because of the change in the conceptual vocabulary for problem solving, set $\mathscr{S}$ must be re-considered, too. With the new objects, different specification statements have to be articulated, and this why set $\mathscr{S}_N$ appears in the definition. However, the designer still wants to pursue the same goals as in the previous frame; therefore, set $S'' \subset \mathscr{S}_N^*$ is formally a translation of the the problem specification into the language of the new frame Φ_{NEW}.

Each of these three auxiliary predicates represents a decision that can be made in different phases of tackling design problem $\mathscr{DP}$. As in chapter 5, all these decisions share one feature: They are all generative operations that change the designer's understanding of the design problem. All these framing predicates deal with relation '$specifies$' and aim to find a new problem specification or a new frame, in which the original specification could be interpreted. The decisions defined above are exploratory probes that need to be proven correct or incorrect by evaluating relation '$satisfies$', which manipulates the problem solving theories, models and solutions. Thus, the mutual interaction of two knowledge sources remains as before.

The breakdown in Figure 9.1 follows similar levels and obeys the same simple rules that were introduced in chapter 5:

1. When using terms 'requirements' and 'constraints', we always mean hard, strict demands that must not be relaxed [Fox94].
2. We do not consider the design problems, where a problem specification can be simplified or relaxed; these issues are covered by others [ZF94].
3. Problem specification may be changed by a monotonic extension of initial sets S and $\mathscr{S}$.

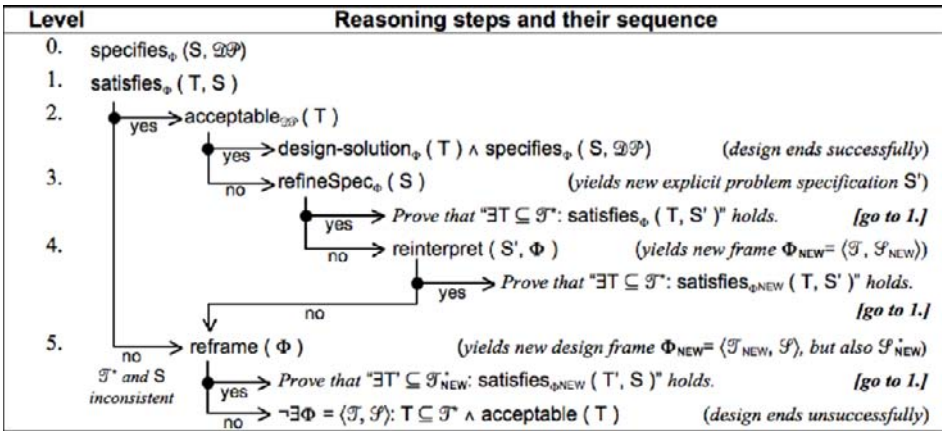


Figure 9.1. Recursive model of framing in design (modifications)

4. The monotonic extension of a problem specification corresponds to fine tuning the problem solving model $T \subset \mathscr{T}$, to narrow down the number of derivable alternatives.
5. Problem specification $S \subset \mathscr{S}$ can be fine-tuned only if an admissible problem-solving model $T \subset \mathscr{T}$ exists for a given design frame Φ.
6. Sentence in the form "Prove that (. . .) holds . . ." represents a recursive step starting always at the level 1. of the recursive model.
7. The recursive step represents a designer's attempt to address given design problem by some modification to the available knowledge sources. It can be understood as an order to evaluate predicate '*satisfies*' *with the (new) arguments provided.*

Let us identify the schemas of reasoning that are explainable from the extended RFD model, and how they correspond to the suggestions made at the beginning of this chapter. These are conceptual schemas, abstract models of reasoning patterns that were observed empirically. They are intended to depict the dependence and sequence of different knowledge-level reasoning steps, as well as the interplay of explicit reasoning strategies with those using the designer's inarticulate knowledge. Schema 2 features only a monotonic refinement of the design frame, whereas schemas 3 and 4 feature two forms of the non-monotonic introduction of new knowledge. Predicate chains in the

schema definition (arrows) refer to the paths between the respective actions (levels) in Figure 9.1.

Reasoning schema 1

$$satisfies_{\Phi}(T,S) \underset{\text{yes}}{\longrightarrow} acceptable_{\mathscr{D}\mathscr{P}}(T) \underset{\text{yes}}{\longrightarrow} designSolution_{\Phi}(T)$$

This schema remains the most trivial design reasoning from the perspective of interplay between articulate and inarticulate knowledge. It was described in chapter 5, and also applies to the modified RFD model. It can be associated with the aggregate results from the experiments, this schema usually models transitions $1a, b \rightarrow 5 \rightarrow 4 \rightarrow 6b$ (that is, satisfy the explicit specification, check that the solution is admissible and refine it if needed). In this case, no tacit knowledge is needed, and the problem can be solved using a formal algorithm. The essence of this schema is thus as follows: "*I know how to define the problem, and I have all necessary data/objects available.*"

Reasoning schema 2

$$satisfies_{\Phi}(T,S) \underset{\text{yes}}{\longrightarrow} acceptable_{\mathscr{D}\mathscr{P}}(T) \underset{\text{no}}{\longrightarrow} refineSpec_{\Phi}(S) \underset{\text{yields}}{\longrightarrow} S' = S \cup \{s \in \mathscr{S}^*\} \underset{\text{yes}}{\longrightarrow} satisfies_{\Phi}(T,S')?$$

This is a situation when, for example, there are multiple solutions available, and all of them are admissible for the given design problem. Although being equally suitable, the designer distinguishes them by their acceptability as design solutions. As defined in chapter 5, an admissible solution is aligned with the designer's implicit and tacit expectations from the product of the design. Since these expectations are not explicitly articulated, there are no criteria that can be used during the admissibility checks. Admissibility has to be replaced by a tacit assessment of acceptability.

If there are multiple solutions and the designer does not accept such an ambiguity, s/he starts looking for a discriminatory condition (requirement, constraint) to separate the acceptable alternatives from the unacceptable ones. The discriminatory condition is taken from the same ontological frame, and its role is to refine the current understanding of the design problem, especially its explicit specification. No external factors are needed to formulate it. This schema

corresponds to a situation when the designer exclaims: "*I have forgotten to say specifically that (... I want the values of parameter X be from range* $\langle Y, Z \rangle$*.)*" In other words, the designer has a conceptual apparatus in the current frame to express the missing statement.

In the terminology of the aggregated results from chapter 8, this schema models sequence of transitions $(5 \rightarrow)7 \rightarrow 3a/6a$; that is, a sound solution exists, but is not acceptable, and the problem specification needs to be refined/extended. After a clarification is made in the specification space, the designer can, in parallel, improve the problem solving model, the solution. This solution refinement corresponds to sequence $(5 \rightarrow)7 \rightarrow 3a/6b.$, but as before, any changes occur inside a single frame, no new concepts are required.

Reasoning schema 3

$$satisfies_{\Phi}(T, S) \underset{\text{yes}}{\longrightarrow} acceptable_{\mathscr{D}\mathscr{P}}(T) \underset{\text{no}}{\longrightarrow} refineSpec_{\Phi}(S) \underset{\text{no}}{\longrightarrow}$$
$$reinterpret(S, \Phi) \underset{\text{yes}}{\longrightarrow} \Phi_{NEW} = \langle \mathscr{T}, \mathscr{S}_N \rangle \underset{\text{yes}}{\longrightarrow}$$
$$\exists T' \subset \mathscr{T}^*, \exists S' \subset \mathscr{S}_N^* : satisfies_{\Phi_{NEW}}(T', S')?$$

Similarly as in schema 2, this is a situation when there is at least one sound solution available but the designer does not accept it. The reasons are similar as earlier, there is a clash between explicit frame $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$ that was used for the problem specification and the solution construction, and the designer's tacit expectations. A logically admissible solution is subjected to an assessment of its acceptability, and as a result a need for the synchronisation of the explicit frame and tacit expectations may emerge. Unlike schema 2, it is not possible to synchronise the tacit expectations using the conceptual vocabulary of the current frame. In other words, the specification set $\mathscr{S}$ does not contain a vocabulary for expressing the new expectations explicitly.

The designer may not be sure how to restore the acceptability, and tries to borrow concepts from a different frame to express his or her expectations in an explicit form. Such a synchronisation is equivalent to a formulation of a new design frame $(\exists \Phi_{NEW} = \langle \mathscr{T}, \mathscr{S}_N \rangle)$ that contains a different vocabulary, which, in turn, may help to reveal missing elements in the original explicit specification of the problem. Thus, the operation of reinterpreting amends first the explicit problem specification primitives $(S' \subset \mathscr{S}_N^*)$, and subsequently, impacts on the process of satisfying the amended specification.

This case is clearly distinct from schema 2 in the fact that a new requirement can be formulated only after shifting the design frame. Without such a shift the requirement remains hidden; in frame Φ there are no conceptual means to express it explicitly. Thus, the following sentence expresses this reasoning. "*I cannot solve the problem specified in this way, something is missing. What if I looked at it from angle X and assumed Y?*" The vocabulary for expressing the missing statement may not be available, but the designer knows how to acquire a new vocabulary.

In the aggregated graphs, this schema models transitions $(5) \rightarrow 7 \rightarrow 8 \rightarrow 3a, 6a$; that is, a frame modification is inserted between the acceptability check and refinement. If a new frame is found while reinterpreting the problem, the designer has to confirm the correctness of the tacit need for a new conceptual frame by implementing the conjectured exploratory probe. New solution $T' \subset \mathscr{T}^*$ needs to be constructed to satisfy the amended specification S'. In the aggregated graphs, this is modelled by transition $7 \rightarrow 8 \rightarrow 6b$, which follows the aforementioned reinterpretation sequence.

Reasoning schema 4

$$satisfies_{\Phi}(T,S) \underset{\text{no}}{\longrightarrow} reframe(\Phi) \underset{\text{yes}}{\longrightarrow} \Phi_{NEW} = \langle \mathscr{T}_N, \mathscr{S}_N \rangle \underset{\text{yes}}{\longrightarrow} \exists T'' \subset \mathscr{T}_N^*, \exists S'' \subset \mathscr{S}_N^* : satisfies_{\Phi_{NEW}}(T'', S'')?$$

Unlike the previous two schemas that occur after the acceptability check, this situation may occur as a result of both, the admissibility or acceptability check. The reason is that the original frame Φ contained mutually incompatible and inconsistent requirements that manifested themselves as an explicit contradiction during the design. The source of the contradiction is in the conceptual foundation of the problem interpretation, in set $\mathscr{T}$. Using that particular conceptualisation and the chosen domain theory DT instantiated by the concepts from $\mathscr{T}$, causes an emergence of a contradiction that could not be resolved by any refinements to the problem specification whether with or without frame re-interpretation.

In other words, if the previous schemas addressed the issue of not having an acceptable problem specification, this schema is looking for the frame in which the problem could be interpreted and tackled. The conceptual vocabulary of frame Φ for the solution construction is incomplete and possibly misleading; to resolve the deadlock new design

frame Φ_{NEW} is articulated. It contains new and different conceptual terms than the old one, and consequently, new, different problem solving theories $\mathscr{T}_N^*$ can be formulated. The novelty of the problem solving theories is emphasised by the opportunity to choose a different generic domain theory DT_{NEW} that fits the new frame better than the previously used domain theories.

New conceptual interpretation of design problem $\mathscr{DP}$ is a non-monotonic and ill-defined operation, but the resolution of the situation is rather well structured, because the designer has an explicit knowledge of a contradiction. There is a (typically constraining) condition that be the focus of the designer's attention whilst searching for a new frame – something that can be assessed for compliance with any new conceptualisation and a conceptual design frame. The following statement expresses this situation: "*I know what is wrong, and I have an idea about how to fix it.*" Similarly as in schema 3, the designer does not have the expressive means in the current frame to formulate those missing concepts; he seeks a new conceptualisation.

In the aggregated graphs, this schema corresponds to transition $(7 \rightarrow)8 \rightarrow 3b, 6b$; after identifying an explicit conflict, define a new conceptual frame and propose new components to form an admissible solution. The reasoning chain may start with an acceptability check; this only emphasises the fact that the discovery of a contradiction may start as an inarticulate feeling that is step by step narrowed to an explicit conflict.

9.2. New design schemas exemplified

Next, the models of different situations covered by the recursive model of framing in design are illustrated with examples. The identified schemas from the previous section are instantiated using the empirical observations acquired during the experiments. The accounts given below are abbreviated from sections 7.2 to 7.5, and focus on the relevant phases that illustrate a particular schema.

Note that any proposed schema may characterise an entire design task. However, in most cases, several schemas appear during the process of designing. This must not be viewed as a shortcoming of the theory of framing in design. The appearance of multiple reasoning schemas in a single design case underlines the ill-structured nature of design problems. It supports the expectation that designers apply a

variety of reasoning techniques, when they are tackling ill-structured design problems. The notion of *multiple paths to a design solution* can be found in [DZ01b, DZ02], an throughout chapters 3 and 4.

Rather than labelling a design task with a unique schema, and following a procedure typically associated with such a schema, design is by nature exploratory. Several paths may lead to an acceptable solution, but it cannot be said in advance, which one will guarantee a results. This is the basis of 'ill structure' [Sim73]; the application of different schemas is also in accordance with the theory of reflection in action [Sch83] – both mentioned several times in chapters 2 to 4.

From the theoretical viewpoint, the intuitively expected coincidence of certain schemas in a design task was confirmed (see chapter 8). For instance, the less structured the design problem is, the higher the frequency of observing schemas 2 or 3, rather than schema 1. Inventive problems exhibit schema 4 more frequently than other schemas. On the other hand, more routine and trivial tasks typically comply with schema 1 and possibly 2. The correlation analyses in section 8.2 confirmed these intuitive expectations rigorously and over a larger number of design sessions.

In the following sections, I present exemplar design scenarios by stating what was the original conceptualisation, domain theory, problem solving theory, problem solving model, design specification, constraining conditions, etc.

9.2.1. Schema 2: design refined by implicit specification

This reasoning schema is commonly observed in design, because designers work with many implicit expectations, in addition to the explicit specification of a design problem. These expectations are not expressed in the same language as the explicit requirements or constraints, and they are known in the cognitive perspective as *design intentions* [NSH94, NYS+98], whereby many intentions remain tacit. Rather than being tact (that is, impossible to articulate), they are usually implicit expectations [DF98].

There is a significant amount of confusion in using terms 'tacit', 'implicit', and 'not explicit'. However, tacitness and explicitness are two different types, two dimensions of knowledge [CB99]; one can be used to acquire another, but it is not possible to turn tacit into explicit. In line with this epistemological perspective the much-discussed process of "surfacing tacit intentions and mapping them

into explicit representations" [NSH94] is mainly about an explicit formulation of implicit intentions. Designers often take some features for granted and do not explicate them unless necessary. Their tacit knowledge, experience may surely help to formulate the untold but important expectations, but the result is only an explicit formulation of a statement in the language of a particular conceptualisation. In other words, such a statement is expressible; it has simply not been expressed explicitly in the previous iterative step.

Consider, for instance, the design situation from section 7.4, where the designer was asked to design a strategy for smoothing raw paper of variable thickness. The problem was initially framed so that the operation of smoothing was associated with the application of pressure, and consequently a pair of rolling drums was suggested as a problem solving model. This approach satisfied the desired requirements on the smoothness and even thickness. The paper was smoothed as intended, however, in some cases, there was an imminent danger of its tearing, which was not desirable. Despite the fact that the prevention of paper tearing was not explicitly articulated in the initial problem specification, it seemed reasonable to add this as another explicit constraint. This explicit commitment reminded the designer that he had to be aware of paper tearing in the future design decisions. The designer justified this extension by pointing to the fact that such a requirement is "so obvious that it is rarely emphasised".

In this articulation of a previously implicit requirement, the current conceptual objects remained unchanged, but the explicit problem specification was refined. This monotonic extension occurred within the original frame that worked with terms as 'rolling', 'pressure', 'drums', etc. However, this simple explication of an implicit requirement had implications on the otherwise non-monotonic problem solving theory. Because of such a refinement, the designer needed to find a new problem solving model to comply with the refined problem specification. In the particular design case, this extension led to the introduction of new structural units to the assembly in the form of 'a moisturiser' and 'a dryer'.

These new conceptual units softened the paper before rolling, so that lower pressure was required, and the danger of tearing was reduced. The actual addition of new components to the solution is a consequence of schema 2, rather than its cause. In this particular case, schema 2 amended the problem specification (transition $7 \rightarrow 6a$), and

was followed by an attempt to satisfy the new specification (transition $7 \rightarrow 3b/6b \rightarrow 5$). The whole episode worked with a single design frame, as follows:

a) **Explicit problem specification** (requirements):
- "Make paper smooth, debris-free";
- "Ensure paper has an even thickness that can be specified"

b) **Problem specification** (constraints):
- "Minimal/maximal pressure for a non-destructive deformation of paper is given by a chart in the catalogue"

c) **Domain theory** (extracts):
- "Theory of mechanics, equations for friction calculation, pressure distribution, material deformation/shaping, etc.";
- "If a maximal strain of the material is exceeded, this may lead to the permanent (non-elastic) destruction of the material"

d) **Conceptual design structures**: (examples)
- "'Rolling drum pair' described by the dimensions of the drums, gap between them, pressure exerted on the material in the gap, etc. (see Figure 9.2)"

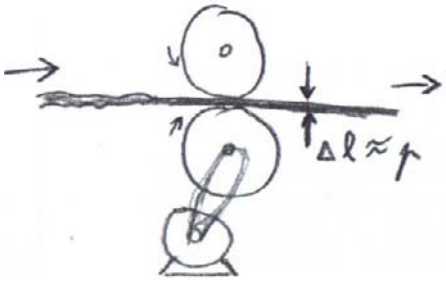


Figure 9.2. Paper-smoothing concept before the implicit refinement

e) **Reason for solution unacceptability**:
- "If we worked with a thinner paper, we could easily exceed the maximal strain threshold"

f) **Refined specification** (previously implicit knowledge):
- "We must avoid the ambiguity, and strictly ensure that paper is not damaged while smoothed";

- "Thus, it seems better to reduce the pressure of the drums significantly below the maximal threshold";
- "But now we need to compensate the loss of pressure"

g) **Newly articulated design solution**:

- "Lack of pressure compensated by loosening the bonds in the material (similarly as with steam ironing or hot metal rolling)";
- "A solution taking refinements into account is in Figure 9.3"

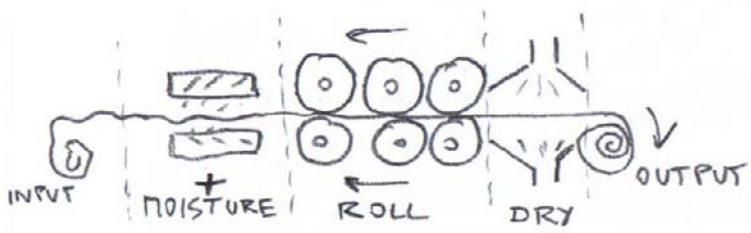


Figure 9.3. Paper-smoothing assembly after the explicated requirement

In the newly articulated constraint (point (f) above), there is no reference to a new concept or a new structural object. Not even the properties mentioned are new to the current conceptual frame. An explicit commitment to a particular feature of the paper that was known in the current conceptual frame is the new knowledge here. Paper continuity and no structural damage were exposed and the designer explicitly articulated, what are the desired values for these properties. Hence, following schema 2, this is a refinement of the problem specification rather than amendment to the design frame.

9.2.2. Schema 3: refinement of problem specification via re-framing

This schema is similar to section 9.2.1; the main difference between these two schemas modifying the problem specification is in the design frame in which the missing statement is formulated. In section 9.2.1 the formulation was achieved in the language available in the current frame. This is not possible in schema 3. In other words, schema 2 is about an explicit commitment to an implicitly present statement. Schema 3 requires more effort to discover what is missing. In the RFD model, this 'extra effort' was modelled as re-framing,

and consequently a formulation of new explicit problem specification of design problem $\mathscr{DP}$.

This is a specific form of frame amendment called *refinement via re-framing*. The actual conceptualisation of the problem, the vocabulary $\mathscr{T}$ does not change. What is changed, is the explicit problem specification. As with schema 2, this is a synchronisation of an explicit frame with tacit expectations and intentions. The refinements modelled by schema 3 were not only forgotten in the original specification, but there was no vocabulary to express them in the first place. This new vocabulary must be brought in from a different frame so that the designer may re-interpret the design problem.

As an illustration, consider the episode from the design of an active shock absorber (section 7.2), where the designer framed the problem with such concepts as 'spring' and 'rigidity'. Early in his design, he discovered that the rigidity of a spring must be modifiable. Rigidity of the shock absorber depends on the toughness of the spring, and to modify the rigidity, this parameter has to change. However, for a common spring it could only be achieved by changing its material or structural properties (such as diameter or length). Therefore, he proposed a functional alternative, 'a pneumatic absorber', and devised a set of rules for amending the pressure inside the piston (Figure 9.4).

It seemed to be a straightforward design from this point onward. However, when reflecting on the proposed solution and the respective controller, the designer became aware of an unexpected behaviour. According to his control strategy, if a disturbance was recorded on the chassis, this was interpreted as a car entering a rough road. The control action immediately softened the absorber and reduced the pressure inside. The stepwise reduction of pressure would stop as soon as the oscillations ceased to be measurable. Then, the designer recalled the frame, from which he borrowed the pneumatic piston, and realised that his control strategy had flaws.

Namely, he became aware of 'one-way control'; he was only able to decrease the pressure when an uneven road was hit. However, this was not a correct interpretation of the term 'active suspension' because it could not perform a reversed effect. Thus, the control needed to be "more active in terms of reacting to the road surface". To that extent, he articulated a need for a reversible pressure control that included a prediction of the future states of the road. The notion of recognising the need to harden the suspension system came in from

an external frame; in the currently worked frame comfort was always linked to softening.

Eventually, new terms extended the explicit problem specification to improve the inadequate understanding of the original problem. However, these terms, requirements and assumptions were not expressible in the original frame. They required the designer to allow for historical data to be used for the prediction. He had to introduce assumptions for the prediction, which were neither requirements nor constraints. These additional statements helped to articulate a more acceptable and more complete specification of the problem, and thus, to refine the understanding of the active shock absorption by referring to the relevant concepts from similar frames (cases).

a) **Explicit problem specification** (requirements):
- "Adjust absorber pressure in response to an uneven surface";
- "Pressure adjustment must be easy to set and fast in response"

b) **Domain theory** (extract):
- "Mechanics, pneumatic systems, pressure equations, IF-THEN productive rules, etc."

c) **Conceptual design structures** (examples):
- "A simple 'shock absorber' consisting of 'a pneumatic piston', 'measuring' and 'control units' as depicted in Figure 9.4";
- "IF the wheel changes its position more than X mm, THEN decrease the pressure of absorber to eliminate the disturbance";
- "IF the chassis oscillation is eliminated quickly, THEN passenger has higher comfort"

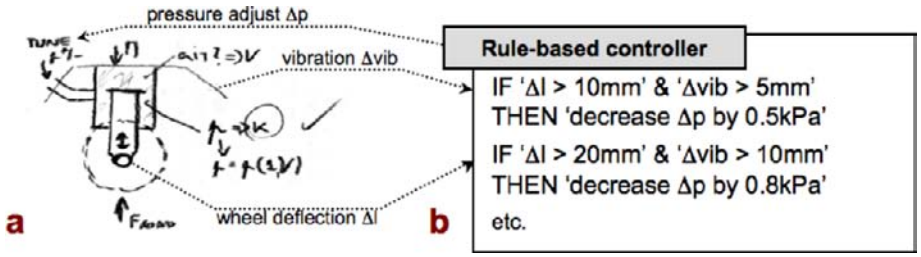


Figure 9.4. A simple shock absorber with a rule-based control strategy

d) **Missing features typically occurring in similar cases**:
   - "The adjustment must be controlled reversibly; be able to increase and decrease pressure (that is, the activity must be more intelligent)"

e) **Tentative proposals:**
   - "If we had some kind of predictive mechanism able of adaptation, it might deliver more active and intelligent control";
   - "What about a new rule: 'IF nothing is happening with the car, THEN keep increasing the pressure to get a stiffer absorber'?";
   - "Such heuristic rules bring in adaptation and a kind of probing how far the controller can go in adjusting the optimality of suspension"

In the extract from the episode, we may see the tentative question "What if?" referring to the specification of the problem but using the terminology of the familiar cases, frames tackled in the past. Similarly as in section 9.2.1, this schema is modelling an extension of the explicit problem specification, but this is a refinement that draws on the tacit discovery of useful features in the past problems and their re-use in the current problem. This reference to conceptually different frames distinguishes schemas 2 and 3. Schema 3 amends the specification rather than generates a new vocabulary for solving the problem. The illustrative scenario corresponds to the sequence of reasoning steps $7 \rightarrow 8 \rightarrow 6a/3a$. It is refining the notion of 'activity' and simultaneously, extending the specification with the concepts 'prediction' and 'adaptation'.

9.2.3. Schema 4: re-framing of a contradictory theory

This schema is typically invoked when the current frame is unable to propose any suitable problem solving model consistent with the required features. The reason may be that some of the specified features are mutually contradictory, and cannot co-exist in the form dictated by the current problem solving theory $\mathscr{T}^*$. The reader remembers that a problem solving theory was an instantiation of a generic domain theory DT using the chosen conceptualisation $\mathscr{T}$. This is not to say that the domain theory is flawed, but clearly, its interpretation in terms of the chosen conceptualisation leads to contradictions.

A contradiction goes deeper than a plain absence of the desired feature in the current state [CRR+90]. The innovative character of

a solution appears in response to challenging these deeply rooted familiar interpretations rather than tackling the absence of the desired, superficial feature. Thus, problems must be seen in a conceptually innovative manner, in order to have innovative solutions.

Let us illustrate this scenario on an episode from the design of a paper-smoothing plant (section 7.4). At a certain stage, the designer identified two contradictory requirements that could not coexist and be satisfied in a single design solution. On one hand, he demanded quality of paper smoothing, expressed by an even thickness and smooth surface. To achieve better quality he added more pairs of rolling drums, and the plant comprised a sequence of drums. This 'extensive approach' to satisfying the quality requirement was logically correct. Nevertheless, adding more drums led nowhere; the design was bound by an implicit constraint on a maximum length of the plant that was linked to the difficulty of controlling it.

It was clear that improving the quality by enlarging the assembly only worsens the maintainability and control-related constraints. The increase in pressure had also its limits, because the higher the pressure, the greater was the danger of tearing the paper. A damaged paper was (obviously) not admissible (see also section 9.2.1). The designer tried to 'squeeze' the rolling drums closer to gain some reserve in the dimensions; however, this was of little use in simplifying the control of the plant. Thus, the designer found himself in a circle of mutually contradicting requirements.

He resolved the deadlock creatively: Instead of squeezing the layout of the rolling assembly in 'one dimension' (linearly laid-out pairs of drums), he amended his conceptual frame. This amendment occurred, when he articulated the concept 'two-dimensional layout' and 'two-dimensional squeezing'. When it was impossible to go beyond the constraining limits in one dimension, he brought in another dimension, and tried the 'squeezing' in two dimensions. The result of this conceptual extension of the design situation was an introduction of the alternately (zigzag) laid-out drums. These had greater effective surface acting on the paper; thus, smaller rolling module was needed, and both size- and pressure-related constraints could be satisfied.

a) **Explicit problem specification** (requirements):
- "Make paper smooth, debris-free";
- "Ensure paper has an even thickness that can be specified";

- "Desired precision of paper thickness is $\pm 0.1mm$"

b) **Explicit problem specification** (constraints):

- "Minimal/maximal pressure for a non-destructive deformation of paper is given by a chart in the catalogue";
- "The rolling section should not have more than three drums to keep it simple"

c) **Domain theory** (extract):

- "Theory of mechanics, equations for friction calculation, pressure distribution, material deformation/shaping, etc.";
- "If a maximal strain of the material is exceeded, this may lead to the permanent (non-elastic) destruction of the material"

d) **Conceptual design structures** (examples):

- "A sequence of 'rolling drum pairs' with a decreasing gap through which the material was passing (see Figure 9.5)"

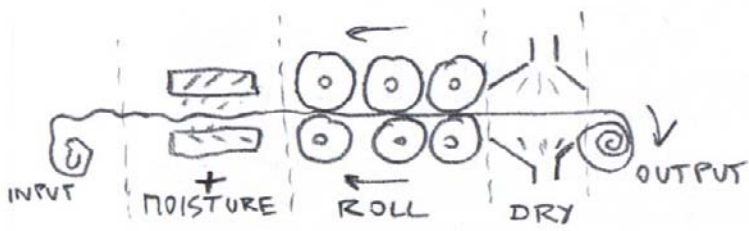


Figure 9.5. Paper-smoothing assembly before the discovery of a contradiction

e) **Explicit contradiction** (extract):

- "In order to improve quality of rolling, we add more drums";
- "More drums may cause problems with maintainability, control";
- "Squeezing of rolling drums does not help";
- "Hence, it is not possible to extend the rolling section and maintain simplicity and vice-versa, it is not possible to shrink it and maintain the quality";

f) **Conceptual resolution of the contradiction** (extract):

- "'Squeezing' the drums in two dimensions instead of one";
- "This would increase the surface interacting with the paper, thus requiring a smaller rolling module and a lower pressure";

g) **Implementation of the conceptual idea** (new structures):
   - "'Zigzag layout' instead of 'linear layout' is actually a squeezing the drums in two dimensions, as illustrated in Figure 9.6"

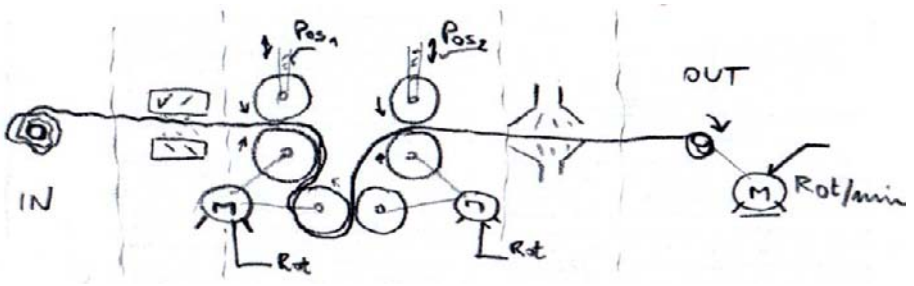


Figure 9.6. Paper-smoothing assembly after the resolution of a contradiction

Re-framing of the design problem is observable in the articulation of the move from 'one-dimensional' design solutions towards 'two-dimensional' products; from the linear layout to the alternate one. This is a change of the frame; a different design frame is introduced to tackle the contradiciton, rather than refining the problem specification. A different conceptual vocabulary is brought in to articulate new objects and relations in multidimensional layout.

Nevertheless, the problem specification remained unchanged; designer still wanted the smooth paper of an even thickness. Only the contradictory conceptual terms were omitted from the new conceptualisation, and were replaced with new, functionally alternative but sound concepts. Unlike schema 3 that only refined the existing problem specification using a new frame, this schema is introducing a new vocabulary for the construction of problem solving models and design solutions. Thus, in this particular case, schema 4 was responsible for a tacit rectification of a contradiction in the problem solving theory, and clearly featured the transition $7 \rightarrow 8 \rightarrow 3b$.

9.3. RFD extensions concluded

This chapter presented a few interesting extensions and refinements to the original RFD model. These particular schemas and patterns of reasoning in design emerged from the directly explainable results of

my experimental study. The extended RFD model is a more accurate description of how the process of design framing and re-framing can be modelled. Nevertheless, this does not mean that this is the final version. In the terminology of my RFD theory, the extended RFD model (see Figure 9.1 earlier in this chapter) is a more acceptable solution to the explanation of the ill-structured design process.

However, it is acceptable from the perspective of an abstract model of a design process. It is far from complete in respect to the practical implementation. There is still a lot of work needed to turn the abstract models and schemas from this research into a practice-ready design support system. Nonetheless, it was not the intention of my research to deliver an off-the-shelf solution to the wicked problem of designing. We sought to acquire a deeper insight and understanding of the process of design (re-)framing on the knowledge level.

Chapter 10

Reflecting on Design and Framing

In the previous chapters I sketched and validated a novel approach to modelling design at the level of knowledge. The model of framing I refer to as *RFD model* represents an interesting formalisation of a grey area in the science of design. In particular, the reasoning schemas exemplified in real examples in chapter 9 represent a contribution to design science at three levels:

- Contributions to the epistemological foundations and design theory;
- Contributions to modelling the inarticulate reasoning processes in design(ing); and
- Contributions to the empirical and methodological research

In addition to discussing value that can be derived from the models presented earlier, I will also mention in this chapter how the theory of framing has been applied in implementing several tools and systems focusing on the user interaction with knowledge modelling and semantic web technologies.

10.1. Implementing design frames in general

The primary point concluding the discussion from the previous chapters relates to the implementation of a computational, knowledgeable tool that realizes the processes defined and formalized in this book. Before discussing the details of how framing was applied in the context of semantic web technologies, let me sketch the principles for any implementations first.

When defining term *design frame* in chapter 5, I mentioned that it was an explicit commitment to a particular, usually familiar, vocabulary. Familiarity suggests the application of techniques for the reasoning by analogy. Analogy (familiarity) can be recognised between a base case (familiar, well-defined previously tackled problem) on one hand, and a target case (current, ill-defined problem) on the other hand. There are several types of familiarities that can be recognised between design cases [GF91], such as abstracted analogy, literal similarity or mere appearance. Let us briefly look at how these types can be used and how they can help with implementing the RFD model. Additional details of reasoning by analogy can be found, for instance, in [FFG89].

Abstracted analogy looks for matches between the higher-level relations among the objects in the similar situations (for example, considering their functions or behaviours), without paying much attention to the low-level details (that is, structures). Since this method recognises familiarity on the level of functions or behaviours, it may discover unusual mappings between the two cases. In session T11 (section 7.2), for example, the designer changed a mechanical spring into a pneumatic piston. Although the two devices are from different domains, they behave similarly. They both tend to counter the external forces acting on them; these counter-actions increase monotonically with the increasing forces. Thus, a pneumatic piston is in a sense, a functionally alternative component to the spring. A new structural concept of piston consequently led to a frame shift in this design session, as explained in section 9.2.3).

The literal similarity is usually justified by the idea that "objects looking alike may as well act alike" [FFG89]. Thus, we may recognise a literal similarity, for instance, between the above-mentioned pneumatic piston (target case) and a hydraulic press (base case). The analogous descriptive parameters of the two cases may then lead to introducing new constraints or assumptions to the explicit specification of the target case. Such a modification of the designer's knowledge was in the RFD theory modelled as a refinement of problem specification within a single frame (section 9.2.1) or with re-framing (section 9.2.2). The two methods for the discovery of similarities complement each other and re-use different knowledge from the base case (structural components for a solution construction as well as functional constraints for a problem specification).

10.2. Design frames and eLearning

One of the aims of the eLearning community over past few years was the development of small structural objects, from which online courses could be composed. These structural objects are known as *learning objects*, and formally, a learning object is any entity, digital or non-digital, that may be used for learning. Functionally, the idea of learning objects aims at learning experiences, where courses and training programmes customised for specific users can be produced quickly, efficiently and economically by choosing and combining standard learning objects from learning repositories. Thus, a lot of emphasis is on making the educational process more 'plug&play' production rather than craft.

However, as as argued by my colleagues [SM04], learning should show three key elements: structure, relatedness and, above all, *interpretation*. Others talk about the form and the relations equally contributing to a successful learning task [Pol03]. Where form or structure in learning corresponds to the engineering objects and modules mentioned in chapter 5, the relatedness can be associated with functional and behavioural paths involving given structural objects. Yet, in addition to appreciating the structure and its relations to the environment, a learner has to interpret and situate his or her learning experience in a fairly complex surrounding world.

The key issue with numerous knowledge portals in general, and with learning object repositories in particular, is that today's portals tent to be built with one hardcoded context in mind. Such contexts are then transformed by a team of knowledge engineers into formal knowledge models or ontologies [Gru93] that are subsequently used by the viewers as referential frameworks for finding content. However, the selection of a particular referential framework for interpreting content and its objects is *consensual* rather than prescriptive. Different stakeholders may want to choose different ontological perspectives to interact with and interpret the same chunk of content.

Thus, one of major challenges in eLearning support is to re-establish a range of different, user-driven contexts for the learning objects residing in various repositories. In our work eLearning was not about finding the content but about understanding how one chunk of content relates to another, and in what particular context does such a relationship makes sense. This line of research is developed in more

details in [DSM05] and other publications; let me focus here on the core of applying the notion of framing.

As defined earlier, a frame is essentially a circumscription of the domain vocabulary, and thus structural primitives one wants to attend to, when interacting with an ill-structured situation. The principle of framing allowed our learners to change their points of view on the fly. Consequently, the same web document could be framed and interpreted from a variety of perspectives. In each particular case, only the underlying conceptual frame (the domain vocabulary and activities available) would differ. For instance, in case of a novice student of climate science, the domain vocabulary (set $\mathscr{S}$) contained the basic concepts about climate (e.g., solar and terrestrial radiation, temperature, or rainfall). Domain glossary for an advanced student was substantially richer and contained concepts from related sciences, too. Terms from physics or chemistry were there to help the student associate the topics specific to the climate science with other subjects – for example, the notion of dark body radiation from physics was linked to the climatological notions of albedo and surface reflection).

The difference between the two frames was even greater in the available learning activities (elements of set $\mathscr{T}$). In this case, a novice gained access to concept explanations or simple indexing services; whereas experts could exploit the acquisition of knowledge about specific experiments, emerging causal relationships or existing scholarly arguments related to a particular conceptual object. Nonetheless, both groups still accessed the same learning object, the same course material. Thanks to framing it differently they were able to experience different learning experiences. Referring back to the vocabulary of traditional learning objects, we see the process of framing as a kind of 'on-the-fly' construction and customization of learning resources into a *unique and transient* learning object. Thus, our "Frame–Navigate–Annotate" paradigm for constructing user facing components supporting content framing emerged as an extension of the theoretical model described earlier in this book.

The main differences of our approach as compared to standard learning object are in the dynamics of the interaction and control over the resources. With regard to dynamics, it is a learner's privilege to choose which particular frame s/he wants to use in a learning situation. With regard to control, learning resources are decentralized

and fully distributed; for instance, basic web document might be created by author A and each of the available learning activities L_i (i=1,2,...) might have (in principle) a different author B_i. It is then through the process of framing and subsequent annotation of the base web document, the learner is able to experience a unified experience – as if the individual learning activities were defined directly in the base document.

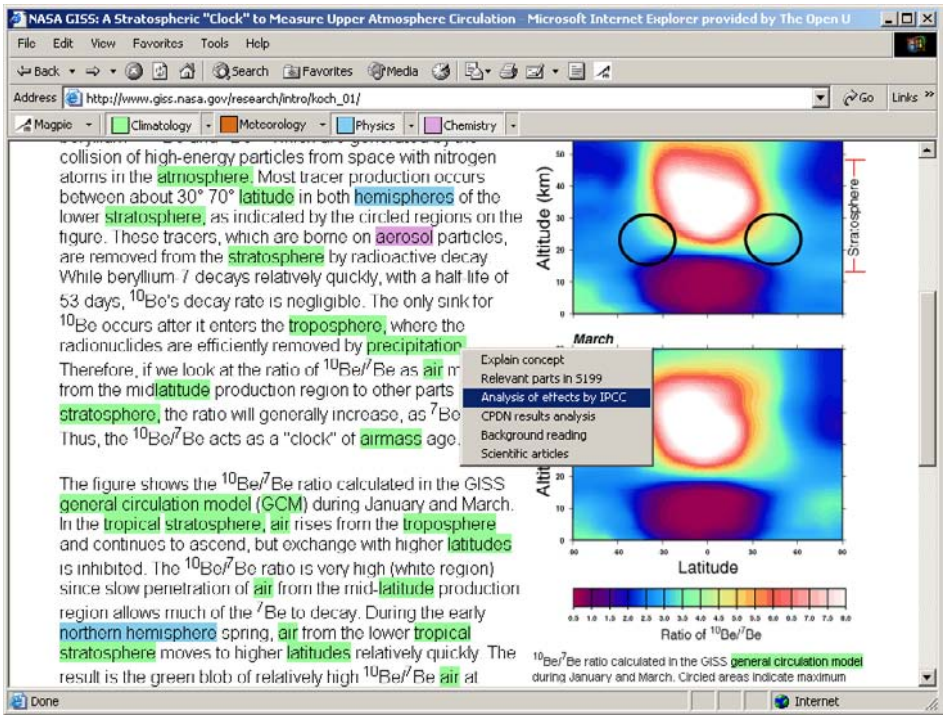


Figure 10.1. A climate science object framed by the climate science ontology and annotated in terms of concepts pertaining to meteorology, climatology, physics and chemistry.

Figure 10.1 shows an initial screenshot of a framed learning object. A customised list of activities next to highlighted word 'precipitation' is a contextually relevant subset of learning and analytic activities applicable to the category of climatological concepts. Unlike traditional portals, the information annotated in the above figure is *soft*; it changes when the ontological frame (that is, the choice of categories in the web browser's toolbar in Figure 10.1) are changed.

The situatedness of learning is achieved by linking the framed and annotated entities to distributed snippets realizing a range of simpler or more complext analytic tasks. The advantage of our 'soft approach' to interacting with learning objects through framing and reframing them is that the users neither need to use a complex query language nor need they browse through an extensive object repository.

Where the interpretative component becomes even more visible are further analyses of the results coming from the interactive services hidden behind the menus shown in Figure 10.1. Instead of merely retrieving cited literature, the about approach allows applying frame-specific links and relationships to the entities the learner is interested in. For instance, one of the words appearing in the figure is *carbon dioxide*. According to a formalised model of a well-known dispute in the climate science, sketched in Figure 10.2, one can draw different argumentative lines depending on whether they believe in the observed increase of carbon dioxide or otherwise.

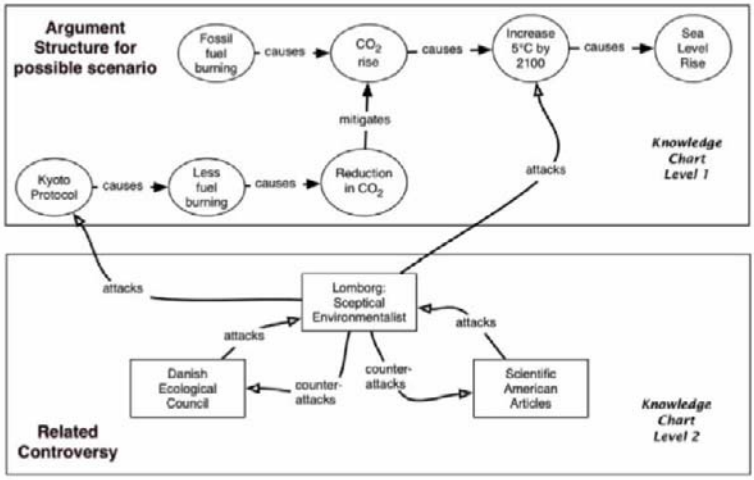


Figure 10.2. Schema of a debate on the role of fossil fuels in climate change involving Lomborg's scepticism and more traditional views on a given climate issue [SM04]

Several causally linked preconditions are suggested that could contribute to either phenomenon. For instance, the '*Kyoto Protocol*' and '*Sea Level Rise*' semantically associate the concept of carbon dioxide with other topics. Two causal diagrams in Figure 10.2 are linked

through a staged argument drawing on additional concepts taken from the investigated frame. As seen in the lower half of Figure 10.2, the core of the dispute is suggested to be in the author's attack on the two independent concepts of the prevailing climate change theory.

In other words, thanks to the initial framing our learner does not only receive an answer but might also develop a skill in associating objects and concepts with each other. Such associations may be *semantic* (e.g., between 'Rise in Temperature' and 'Global Warming'), *causal* (e.g., between 'Rise in Temperature' and 'Rise in Sea Level'), *exclusive* (e.g., either 'CO_2 Rise' or 'CO_2 Reduction'), *supportive* (e.g., between 'European Commission Initiative' and 'Kyoto Protocol'), or *adversarial* (e.g., between 'Sceptical Environmentalism' and 'Catastrophic Environmentalism').

10.3. Framing and semantic web browsing

In parallel with researching the role of framing in eLearning, I also investigated a more general problem – how to overcome the knowledge acquisition bottleneck [HRWL83] in the context of using ever increasing pool of web resources. To address this bottleneck since finishing the theoretical work I started developing demonstrations of Semantic Web-like capabilities [BLHL01] – in the absence of the key ingredient, the semantic markup. Between 2003 and 2008 I led the development of *Magpie*, a suite of tools supporting low-cost annotation of web documents and their browsing using the semantic links. The key feature of the Magpie line that distinguished it from previous attempts to browse Semantic Web, is its capability to support both the annotation and subsequent *interpretation* of web documents with no a-apriori markup [DMD07].

Following on the educational scenario described in section 10.2, several other Magpie-based applications were developed. In the context of climate science demonstrator Magpie was seen as a tool supporting the interpretation of web pages. The recognition of entities and their annotation based on a user-chosen ontological frame were emphasised as means bringing the interpretative context to bear into any web-accessible web page. The main functional achievement of these early implementations was to use a semantically enriched web browser as a tool through which a lay learner may adopt the viewpoint of an expert in climatology.

Through the stage of a generic semantic web browser I realized a different implementation of the theory of framing in the shape of a generic shell for developing semantic web applications [DMD04]. At this evolutionary stage the Magpie suite provided generic mechanisms to apply ontologies as most common conceptual frames in enriching user experiences on the web. The key feature of Magpie in the role of a framework was that it allowed the developers to focus on the actual frames, on the semantic functionalities of their applications, that is on choosing, specifying and populating domain ontologies that together formed a particular semantic application. Much of the user interaction with ontological frames was taken care by the actual Magpie infrastructure and web browser plugins.

When analysing the benefits and costs of using Magpie suite in various contexts [DMD07], I identified several key features that, in my opinion, affected the success of semantic web browsers. While these observations are formulated in the language of semantic web community, one can easily draw links to the claims made in earlier parts of this book about framing and designing in general. For example, my key recommendation to the community was to not only see ontologies as abstract knowledge 'buckets' but treat them as a key component enabling grounding a particular user-facing functionality in some conceptual frame. Even better, semantic web browsers were required to cope with multiple ontologies being applied to the same resource (read 'multiple frames').

Without this explicit commitment to a specific (albeit possibly user-chosen) frame, there is no inherent meaning in mere annotations of web resources, unless the user decides to commit to a particular interpretative frame – domain ontology. Moreover, as illustrated in Magpie, although ontologies are useful, they are abstract entities. In order to make use of them, one should treat them as frames, in the sense defined in this book. In other words, one needs to bring in concrete concepts, object, relations and roles that are implied by the selection of a particular ontological frame, and apply those concrete entities to the interpretation of the web resources. In the language of my theory, this corresponds to the move from abstract sets $\mathscr{T}$ and $\mathscr{S}$ to sufficient specifications $S \subset \mathscr{S}^*$ and $T \subset \mathscr{T}^*$.

Pushing this perspective further, in the real world, it is unlikely that anybody would use only one ontological frame or perspective while interacting with a web browser. Translated to Magpie and to

the theory of framing this means the capability to work with any number of frames simultaneously. Not only that, in the context of semantic web it is unlikely that all ontologies would be accessible from the same location – on the contrary, they are distributed, networked and interlinked. Given this, it is no longer sufficient to allow the user to choose their ontological frame. Users may want to create their personalized frames and viewpoints by mashing together various existing ontological frames.

One area where I see my research going is towards broadening and further loosening of the interfaces between the low-level structural components. One direction where my Magpie research may be heading is the development of a kind of semantic web gateway, an entry point from which the user may start exploring various knowledge-level consequences and functionalities. To this extent, together with one of my research students I have started work on the next generation of semantic web browsers, with a working name 'PowerMagpie'. This prototype, shown in Figure 10.3 draws upon a newly developed semantic web data gatherer Watson that provides access to ontologies that may be relevant to the given web resource.

Once ontologies are found, PowerMagpie is able to create from these multiple perspectives a single, unique frame which it literally applies to the web resource in question. The outcome is a web resources 'dressed' in a unique semantic skin, which can then be structured according to themes, individual chunks of knowledge found, general object classes, and so on. As Figure 10.3 shows, the move to parametrise a web browser with multiple ontologies is indeed realistic and is likely to increase the power of the users in interpreting the web resources. This latest incarnation of the early ideas of problem framing in terms of its specification and solution entities $\mathscr{T}$ and $\mathscr{S}$ is a direct descendant of the theory described in this book.

Of course, there is still a lot to be done with framing in design and framing in general. I hope this book may act as a good starting point for you, the readers, in your pursuit of new, even more powerful applications that offer the problem framing capability to the end users. I am looking forward to reading about your ideas of problem framing applied to your domain...

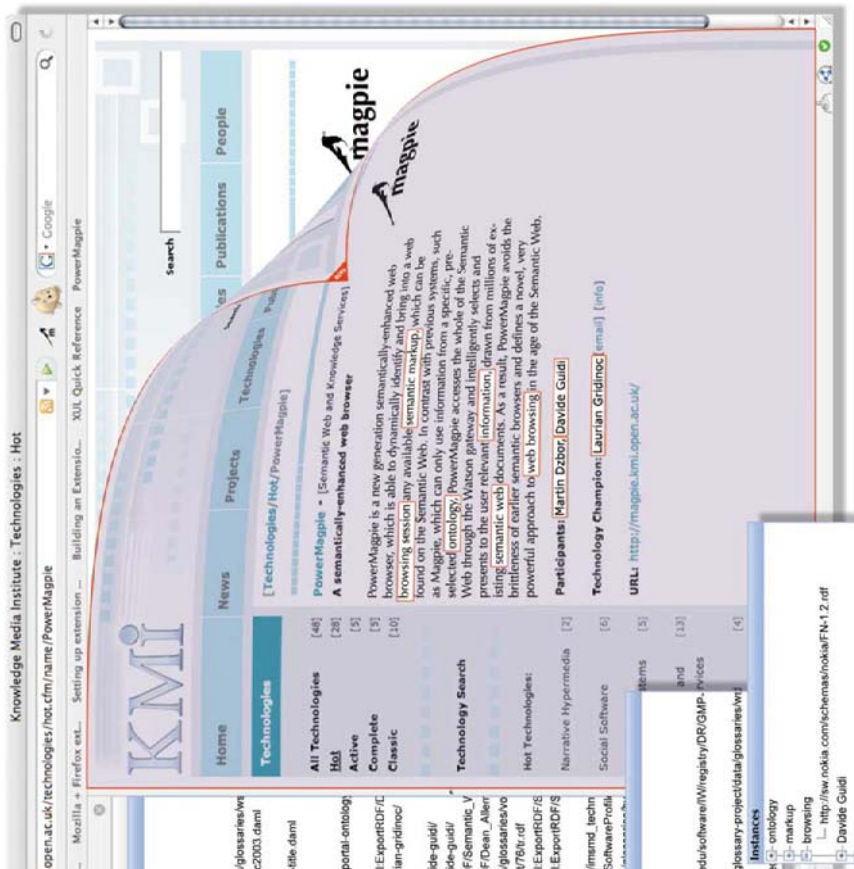


Figure 10.3. A snapshot of PowerMagpie embodying some of the visions about problem framing in the context of semantic web browsing

Bibliography

[AC93] W.L. Anderson and W.T. Crocca. Engineering practice and co-development of product prototypes. *Communications of the ACM*, 36(6):49–56, 1993.

[ACM93] ACM. Special issue on participatory design. *Communications of the ACM*, 36(6), 1993.

[Alt84] G.S. Altshuller. *Creativity as an Exact Science*, volume 5 of *Studies in Cybernetics*. Gordon & Breach Science Publishers, USA, 1984.

[Asi62] M. Asimow. *Introduction to design*. Prentice-Hall, Englewood Cliffs, NJ, 1962.

[BB97] D.C. Brown and W.P. Birmingham. Understanding the nature of design. *IEEE Intelligent Systems & their applications*, 12:14–16, 1997.

[BC85] T. Bylander and B. Chandrasekaran. Understanding behavior using consolidation. In *9th Intl. Joint Conference on AI (IJCAI'85)*, volume 1, pages 450–454, Los Angeles, California, 1985.

[BF82] A. Barr and E.A. Feigenbaum. *The Handbook of Artificial Intelligence*, volume 1. William Kaufmann, Inc., California, USA, 1982.

[BG94] S. Bhatta and A. Goel. Discovery of physical principles from design experiences. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 8(2), 1994.

[BGP94] S. Bhatta, A. Goel, and S. Prabhakar. Innovation in analogical design: A model-based approach. In *3rd Intl. Conference on AI in Design (AID'94)*, pages 55–74, Lausanne, Switzerland, 1994.

[Bla99] P.E. Black. Dictionary of algorithms, data structures, and problems, 1999.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 279(5):34–43, 2001.

[Bor97] W.N. Borst. *Construction of engineering ontologies for knowledge sharing and reuse*. Ph.d thesis (no. 97-14), 1997.

[Boy89] J-M. Boyle. Interactive engineering systems design: a study for artificial intelligence applications. *Artificial Intelligence in Engineering*, 4(2):58–69, 1989.

[BvLT+96] F.M.T. Brazier, P.H.G. van Langen, J. Treur, N.J.E. Wijngaards, and M. Willems. Modelling an elevator design task in desire: the vt example. *Int. Journal of Human-Computer Studies*, 44(3):469–520, 1996.

[BvLT98] F.M.T. Brazier, P.H.G. van Langen, and J. Treur. Strategic knowledge in design: a compositional approach. *Knowledge-based Systems*, 11(Special issue):405–416, 1998.

[BvLTW96] F.M.T. Brazier, P.H.G. van Langen, J. Treur, and N.J.E. Wijngaards. Redesign and reuse in compositional knowledge-based systems. *Knowledge-based Systems*, 9:105–118, 1996.

[Can98] L. Candy. Representations of strategic knowledge in design. *Knowledge-Based Systems*, 11:379–390, 1998.

[CB99] S.D.N. Cook and J.S. Brown. Bridging epistemologies: The generative dance between organizational knowledge and organizational knowing. *Organization Science*, 10(4):381–400, 1999.

[CE96] L. Candy and E. Edmonds. Creative design of the lotus bicycle: implications for knowledge support systems research. *Design Studies*, 17:71–90, 1996.

[CF82] P.R. Cohen and E.A. Feigenbaum. *The Handbook of Artificial Intelligence*, volume 3. William Kaufmann, Inc., California, USA, 1982.

[Cha90] B. Chandrasekaran. Design problem solving: A task analysis. *AI Magazine*, 11(4):59–71, 1990.

[Cha93] B. Chandrasekaran. Functional representation: A brief historical perspective. *Applied Artificial Intelligence*, 1993.

[CJB99] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them. *IEEE Intelligent Systems & their applications*, 14(1):20–26, 1999.

[CMI+98] B.Y. Choueiry, S. McIlraith, Y. Iwasaki, T. Loeser, T. Neller, R.S. Engelmore, and R. Fikes. Thoughts towards a practical theory of reformulation for reasoning about physical systems. Technical Report KSL-98-18, Knowledge Systems Laboratory, 1998.

[Cro97] N. Cross. Descriptive models of creative design: application to an example. *Design Studies*, 18:427–440, 1997.

[Cro99] N. Cross. Natural intelligence in design. *Design Studies*, 20(1):25–39, 1999.

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge-based Design Systems*. Adison-Wesley, 1990.

[CvB01] X. Chen and P. van Beek. Conflict-directed backjumping revisited. *Journal of Artificial Intelligence Research*, 14:53–81, 2001.

[Dav80] M. Davis. The mathematics of non-monotonic reasoning. *Artificial Intelligence*, 13(1):73–80, 1980.

[Dav95] S.P. Davies. Effects of concurrent verbalization on design problem solving. *Design Studies*, 16:102–116, 1995.

[DD95] R.L. Dominowski and P. Dallob. Insight and problem solving. In R.J. Sternberg and J.E. Davidson, editors, *The Nature of Insight*, pages 33–62. MIT Press, USA, 1995.

[Dec92] R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopaedia of Artificial Intelligence*, pages 276–285. John Wiley & Sons, 2nd edition, 1992.

[DF98] Z. Dienes and R. Fahey. The role of implicit memory in controlling a dynamic system. *The Quarterly Journal of Experimental Psychology*, 51A(3):593–614, 1998.

[dGRNK99] A.S. d'Avila Garcez, A. Russo, B. Nuseibeh, and J. Kramer. Restructuring requirements specifications for analysis and change management. In *16th IEEE Conference on Automated Software Engineering*, pages 354–358, San Diego, California, 1999.

[dK86a] J. de Kleer. An Assumption-based TMS. *Artificial In-*

telligence, 28(2):127–162, 1986.

[dK86b] J. de Kleer. Extending the ATMS. *Artificial Intelligence*, 28(2):163–196, 1986.

[dKMR90] J. de Kleer, A.K. Mackworth, and R. Reiter. Characterizing diagnoses. In *AAAI'1990*, volume 1, pages 324–330, 1990.

[DMD04] M. Dzbor, E. Motta, and J. Domingue. Opening up magpie via semantic services. In *In Proc. of the 3rd Intl. Semantic Web Conf.*, 2004.

[DMD07] M. Dzbor, E. Motta, and J. Domingue. Magpie: experiences in supporting semantic web browsing. *Journal of Web Semantics*, 5(3):204—222, 2007.

[Doy79] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[DSM05] M. Dzbor, A. Stutt, and E. Motta. Achieving higher-level learning through adaptable semantic web applications. *Intl. Journal of Knowledge and Learning*, 1(1/2):25–43, 2005.

[DZ01a] M. Dzbor and Z. Zdrahal. Towards a framework for acquisition of design knowledge. In *27th Design Automation Conference, (part of ASME DETC)*, pages DETC01/DAC–21049, Pittsburgh, USA, 2001.

[DZ01b] M. Dzbor and Z. Zdrahal. Towards logical framework for sequential design. In *13th Intl. Conf. on Design Theory and Methodology, (part of ASME DETC)*, pages DETC01/DTM–21710, Pittsburgh, USA, 2001.

[DZ02] M. Dzbor and Z. Zdrahal. Design as interactions between problem framing and problem solving. In *15th European Conference on AI (ECAI)*, pages 210–214, France, 2002.

[Dzb00a] M. Dzbor. Design as a problem of requirements explication. In *ECAI'2000 Workshop on Knowledge-Based Systems for Model-Based Engineering Design*, Berlin, Germany, 2000.

[Dzb00b] M. Dzbor. Explication of design requirements through reflection on solutions. In *4th IEEE Conf. on Knowledge-based Intelligent Engineering Systems*, pages 141–144, Brighton, UK, 2000.

[Dzb01] M. Dzbor. Knowledge re-formulation in design. In N. Baba, L.C. Jain, and R.J. Howlett, editors, *5th*

Conf. on Knowledge-based Intelligent Engineering Systems, volume 1, pages 191–195, Osaka, Japan, 2001. IOS Press/Ohmsha.

[Eze00] M.L. Ezekiel. Just-in-time design in a fast-paced product group. In *Designing Interactive Systems Conference*, New York, USA, 2000.

[FFG89] B. Falkenhainer, K. Forbus, and D. Gentner. The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1–63, 1989.

[Fis92] G. Fischer. Domain-oriented design environments. In *7th Knowledge-Based Software Engineering Conference (KBSE'92)*, pages 204–213. IEEE Computer Society, 1992.

[FNO93] G. Fischer, K. Nakakoji, and J. Ostwald. Facilitating collaborative design through representations of context and intent. In *AAAI Äô93, AI in collaborative design*, Washington, 1993.

[Fox94] M.S. Fox. ISIS: A Retrospective. In M. Zweben and M.S. Fox, editors, *Intelligent Scheduling*, pages 3–29. Morgan Kaufmann Publishers, Inc., 1994.

[G̈30] K. Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik*, 37:349–360, 1930.

[Gar98] J.H. Garrett. The computer-aided engineer: Prospects and risks. *AI for Engineering, Design, Analysis and Manufacturing*, 12:61–63, 1998.

[GdSGM96] A. Gomez de Silva Garza and M.L. Maher. Design by interactive exploration using memory-based techniques. *Knowledge-Based Systems*, 9(1), 1996.

[Ger90] J.S. Gero. Design prototypes: A knowledge representation schema for design. *AI Magazine*, 11(4):26–36, 1990.

[Ger96] J.S. Gero. Creativity, emergence and evolution in design. *Knowledge-Based Systems*, 9:435–448, 1996.

[Ger98a] J. Gero. Concept formation in design. *Knowledge-Based Systems*, 11:429–435, 1998.

[Ger98b] J.S. Gero. Conceptual designing as a sequence of situated acts. In I. Smith, editor, *Artificial Intelligence in Structural Engineering*, Lecture Notes in AI, pages 165–177. Springer, Berlin, 1998.

[GF91] D. Gentner and K. Forbus. MAC/FAC: A Model of Similarity-based Retrieval. In *13th Annual Conference of the Cognitive Science Society*, pages 504–509, Chicago, USA, 1991.

[GI91] T. R. Gruber and Y. Iwasaki. How things work: Knowledge-based modelling of physical devices. Technical Report KSL-90-51, Stanford University, 1991.

[GN87] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publ., USA, 1987.

[Goe94] V. Goel. A comparison of design and nondesign problem spaces. *Artificial Intelligence in Engineering*, 9(1):53–72, 1994.

[Goe95] V. Goel. *Sketches of Thought*. MIT Press, Massachussetts, USA, 1995.

[Goe97] A.K. Goel. Design, analogy, and creativity. *IEEE Expert*, 12(3):62–70, 1997.

[GR91] T. R. Gruber and D.M. Russell. Design knowledge and design rationale: A framework for representation, capture and use. Technical Report KSL-90-45, Stanford University, 1991.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–221, 1993.

[HRWL83] F. Hayes-Roth, D.A. Waterman, and D.B. Lenat. *Building Expert Systems*. Addison-Wesley, Massachusetts, US., 1983.

[IFVC93] Y. Iwasaki, R. Fikes, M. Vescovi, and B. Chandrasekaran. How things are intended to work: Capturing functional knowledge in device design. In *13th IJCAI*, pages 1516–1522, 1993.

[JRC+00] P. Jagodzinski, F.J.M. Reid, P. Culverhouse, R. Parsons, and I. Phillips. A study of electronics engineering design teams. *Design Studies*, 21(4):375–402, 2000.

[Kli85] G.J. Klir. *Architecture of Systems Problem Solving*. Plenum Press, New York, USA, 1985.

[KMM93] F. Kensing and A. Munk-Madsen. PD: Structure in the Toolbox. *Communications of the ACM*, 36(6):78–85, 1993.

[Kum92] V. Kumar. Algorithms for constraints satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[KW96] J.L. Kolodner and L.M. Wills. Powers of observation in creative design. *Design Studies*, 17:385–416, 1996.

[LCS92] B. Logan, D. Corne, and T. Smithers. The Edinburgh Designer System: An Architecture for Solving Ill Structured Problems. In *European Conference on Artificial Intelligence (ECAI)*, Vienna, Austria, 1992.

[Lev74] M. Levin. Mathematical logic for computer scientists. Technical Report MAC project TR-131, MIT, 1974.

[Lev89] H. Levesque. A knowledge-level account of abduction. In *11th IJCAI*, pages 1061–1067, Detroit, USA, 1989.

[Liu00] Y.T. Liu. Creativity or novelty? *Design Studies*, 21(3):261–276, 2000.

[LS92] B. Logan and T. Smithers. Creativity and design as exploration. DAI Research Paper 597, DAI, University of Edinburgh, 1992.

[Mac90] K.J. MacCallum. Does Intelligent CAD exist? *Artificial Intelligence in Engineering*, 5(2):55–64, 1990.

[MC02] M. Muller and K. D. Carey. Design as a minority discipline in a software company: Toward requirements for a community of practice. In *Computer-human interaction (SIGCHI)*, Minnessota, USA, 2002.

[McC80] J. McCarthy. Circumscription – a form of nonmonotonic reasoning. *Artificial Intelligence*, 13(1):27–39, 1980.

[McC86] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[MD80] D. McDermott and J. Doyle. Non-monotonic logic. *Artificial Intelligence*, 13(1-2):41–72, 1980.

[Men79] E. Mendelson. *Introduction to Mathematical Logic.* D. Van Nostrand Co., USA, 2nd edition, 1979.

[Mil93] S.E. Miller. From system design to democracy. *Communications of the ACM*, 36(6), 1993.

[ML84] J. Mylopoulos and H.J. Levesque. An overview of knowledge representation. In M.L. Brodie, J. Mylopoulos, and J.W. Schmidt, editors, *On Conceptual Modelling*, pages 3–17. Springer Verlag, New York, USA, 1984.

[Mot97] E. Motta. *Reusable Components for Knowledge Mod-*

elling. Frontiers in AI and Applications. IOS Press, The Netherlands, 1997.

[Mul02] M. Muller. Participatory Design: The Third Space in HCI. In J. Jacko and A. Spears, editors, *Human Factors And Ergonomics: Handbook of HCI*, pages 1051–1068. L. Erlbaum Associates Inc. New Jersey, USA, 2002.

[MZ96] E. Motta and Z. Zdrahal. Parametric design problem solving. Technical report KMi-TR-26, KMi, The Open University, 1996.

[MZ98] E. Motta and Z. Zdrahal. A principled approach to the construction of a task-specific library of problem solving components. In *11th Banff Knowledge Acquisition for KBS Workshop*, volume 1, Canada, 1998.

[NCB97] S. Nidamarthi, A. Chakrabarti, and T.P. Bligh. The significance of co-evolving requirements and solutions in the design process. In *11th ICED*, Finland, 1997.

[NE00] B. Nuseibeh and S. Easterbrook. Requirements engineering: A roadmap. In *International Conference on Software Engineering*, Limerick, Ireland, 2000.

[New82] A. Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.

[NS76] A. Newell and H. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[NSH94] K. Nakakoji, T. Sumner, and B. Harstad. Perspective-based critiquing: helping designers cope with conflicts among design intentions. In Gero and Sudweeks, editors, *AI in DesignÄô94*, pages 449–466, 1994.

[Nus01] B. Nuseibeh. Weaving together requirements and architectures. *IEEE Computer*, 34(3):115–117, 2001.

[NYS+98] K. Nakakoji, Y. Yamamoto, T. Suzuki, S. Takada, and M.D. Gross. From critiquing to representational talkback: Computer support for revealing features in design. *Knowledge-Based Systems*, 11:457–468, 1998.

[Oxm90] R. Oxman. Design shells: a formalism for prototype refinement in knowledge-based design systems. *Artificial Intelligence in Engineering*, 5(1):2–8, 1990.

[PG98] S. Prabhakar and A. Goel. Functional modeling for enabling adaptive design for new environments. *Artificial*

Intelligence in Engineering, 12:417–444, 1998.

[PL88] C. Price and M. Lee. Applications of deep knowledge. *Artificial Intelligence in Engineering*, 3(1):12–17, 1988.

[Pol03] P.R. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, 3(4):164, 2003.

[Poo89a] D. Poole. Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence*, 5(2):97–110, 1989.

[Poo89b] D. Poole. Normality and faults in logic-based diagnosis. In *11th IJCAI*, pages 1304–1310, Detroit, USA, 1989.

[Poo90] D. Poole. A methodology for using a default and abductive reasoning system. *Intl. Journal of Intelligent Systems*, 5(5):521–548, 1990.

[QG96] L. Qian and J.S. Gero. Function-Behaviour-Structure Paths and Their Role in Analogy-Based Design. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 10:289–312, 1996.

[Red88] M.J. Reddy. The conduit metaphor - a case of frame conflict in our language about language. In A. Ortony, editor, *Metaphor and Thought*, pages 284–324. Cambridge University Press, 1988.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1):81–132, 1980.

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[RNK99] A. Russo, B. Nuseibeh, and J. Kramer. Restructuring requirements specifications for analysis and change management. *IEE Proceedings: Software*, 146(1):44–53, 1999.

[RS89] C. Riesbeck and R. Schank. *Inside Case-based Reasoning*. Lawrence Erlbaum, New Jersey, 1989.

[RSP95] A. Rose, B. Shneiderman, and C. Plaisant. An applied ethnographic method for redesigning user interfaces. Report 95-07, University of Maryland, 1995.

[SB82] W. Swartout and R. Balzer. On the inevitable intertwining of specification and implementation. *Communications of the ACM*, 25(7):438–440, 1982.

[SBHK97] T. Sumner, N. Bonnardel, and B. Harstad Kallak. The cognitive ergonomics of knowledge-based design support

system. In *Computer-Human Interfaces 1997*, pages 83–90, Atlanta, Georgia, 1997.

[SBS98] T. Sumner and S. Buckingham Shum. From documents to discourse: Shifting conceptions of scholarly publishing. In *Conference on Computer-Human Interaction (CHI'1998)*, Los Angeles, California, 1998.

[SCD+90] T. Smithers, A. Conkie, J. Doheny, B. Logan, K. Millington, and M.X. Tang. Design as intelligent behaviour: an AI in design research programme. *Artificial Intelligence in Engineering*, 5(2):78–109, 1990.

[Sch83] D.A. Schön. *Reflective Practitioner – How professionals think in action*. Basic Books, Inc., USA, 1983.

[Sch95] D.A. Schön. Knowing-in-action: The new scholarship requires a new epistemology. *Change - magazine of higher learning*, 27(November/December 1995):27–34, 1995.

[SF94] D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *2nd Intl. Workshop on Principles and Practice of Constraint Programming*, pages 10–20, Washington, USA, 1994.

[Shu99] L. Shulyak. Introduction to TRIZ, 1999.

[Sim69] H.A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, 1969.

[Sim73] H.A. Simon. The structure of ill-structured problems. *Artificial Intelligence*, 4:181–201, 1973.

[SM04] A. Stutt and E. Motta. Semantic webs for learning: A vision and its realization. In *In Proc. of the 14th European Conf. on Knowledge Engineering and Management (EKAW)*, pages 132–143, 2004.

[SMW95] V.V. Sushkov, N.J.I. Mars, and P.M. Wognum. Introduction to TIPS: a theory for creative design. *Artificial Intelligence in Engineering*, 9(2):177–189, 1995.

[SSS+01] R. Sara, M. Svec, D. Smutek, P. Sucharda, and S. Svacina. Texture analysis of sonographic images for hashimoto's lymphocytic thyroiditis recognition, 2001.

[Ste94] L. Steinberg. Research methodology for ai and design. *AI for Engineering Design, Analysis, and Manufacturing*, 8:283–287, 1994.

[Ste95] M. Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann, California, 1995.

[STS+01] D. Smutek, T. Tjahjadi, R. Sara, M. Svec, P. Sucharda, and S. Svacina. Image texture analysis of sonograms in chronic inflammations of thyroid gland. Report CTU-CMP-2001-15, Czech Technical University, 2001.

[Tan97] M. Tang. A knowledge-based architecture for intelligent design support. *Knowledge Engineering Review*, 12(4):387–406, 1997.

[TH93] D.S.W. Tansley and C.C. Hayball. *Knowledge-based systems analysis and design*. BCS Practitioner. Prentice Hall International (UK) Ltd., 1993.

[TL99] G. Thompson and M. Lordan. A review of creativity principles applied to engineering design. *Journal of Process Mechanical Engineering*, 213(1):17–31(15), 1999.

[TN94] H. Takeda and T. Nishida. Integration of aspects of design processes. In *AID'1994*, pages 309–326, 1994.

[Tom94] T. Tomiyama. From general design theory to knowledge-intensive engineering. *AI for Engineering Design, Analysis, and Manufacturing*, 8:319–333, 1994.

[Tuf01] E. Tufte. Presenting data and information, 2001.

[Ull97] D.G. Ullman. *The Mechanical Design Process*. McGraw-Hill, Inc., 2 edition, 1997.

[WAS95] B. Wielinga, J.M. Akkermans, and A.T. Schreiber. A formal analysis of parametric design problem solving. In *9th Knowledge Acquisition for KBS Workshop*, pages 37:1–15, Banff, Canada, 1995.

[Wev99] J.C.A. Wevers. Physics formulary. Available from http://www.xs4all.nl/johanw, 1999.

[WP97] I. Watson and S. Perera. Case-based design: A review and analysis of building design applications. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 11(1):59–87, 1997.

[Yos92] G.R. Yost. Configuring elevator systems. Technical report, Digital Equipment Corporation, 1992.

[Zdr97] Z. Zdrahal. DccS truck cabin design, 1997.

[ZF94] M. Zweben and M.S. Fox. *Intelligent Scheduling*. Morgan Kaufmann Publishers, Inc., 1994.

This page intentionally left blank

Index

Appendices

This page intentionally left blank

Appendix A: Design episodes (transcription)

T11: Control of active shock absorption

Design brief

Suspension and shock absorption are crucial elements in the car design, particularly with regard to the passengers comfort. Design an active shock absorber and suggest a control strategy, so that the whole suspension mechanism would be able to adjust its properties according to the current state of the road. In addition to the absorption of shocks caused by an uneven road, we would like to adjust also the chassis clearance, so that the car has the optimal aerodynamic coefficients for a particular type of surface.

- **Context number 2 (DC-II)**
  Articulated requirements (objectives):
  - **Req-1. Control of shock absorption**: The system should be able to control the car suspension/shock absorption.
    - **Req-1.1 Automatically adjustable shock absorption**: Moreover, the absorption must be automatically adjustable in response to the actual terrain, road surface.
  - **Req-2. Chassis clearance control**: The system should be also able to work with chassis clearance – height above road.
    - **Req-2.2 Manually adjustable clearance**: Let's take first the simpler case, when chassis clearance is constant, and can only change in response to drivers explicit request; e.g., off-road, motorway.
  - **Req-3. Need to measure the deflection of the wheel**: In order to control the shock absorbers behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation
  - **Req-4. Need to measure also the vibrations**: OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration
  - **Req-5. Shock absorber rigidity must be adjustable**: This is essential to have an 'active' absorber, the shock absorbing properties must be adjustable in response to road conditions. Moreover, this should be easy to do!
  - **Req-6. Controller for the shock absorption**: See scheme D as an application of a simple control loop that can be used in this case.

Articulated solutions (solution models):

- **Sol-1. Simple control of the shock absorbers**: See scheme D., so far we worked with a constant clearance that can't be changed.

- **Context number 3 (DC-III)**

Articulated requirements (objectives):

- **Req-1. Control of shock absorption**: The system should be able to control the car suspension/shock absorption.
  - **Req-1.3 Automatically AND manually adjustable absorption** (*new*): This is a better alternative, the driver should be able to choose himself if he wants auto-mode or manually sets up the absorber rigidity, toughness.
- **Req-2. Chassis clearance control**: The system should be also able to work with chassis clearance – height above road.
  - **Req-2.3 Automatically and manually adjustable clearance** (*new*): Similarly, the driver should be able to say if he wants an automated optimisation of chassis clearance or sets the value manually to address a specific request.
- **Req-3. Need to measure the deflection of the wheel**: In order to control the shock absorbers behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation
- **Req-4. Need to measure also the vibrations**: OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration
- **Req-5. Shock absorber rigidity must be adjustable**: This is essential to have an 'active' absorber, the shock absorbing properties must be adjustable in response to road conditions. Moreover, this should be easy to do!
- **Req-7. Controlling the clearance** (*new*): OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.
- **Req-8. Joint controller of absorption AND clearance** (*new*): This is a kind of 'master' controller that will coordinate the actions of the two simpler or dedicated sub-controllers. This controller will also be responsible for translating the driver's choices and commands to the values suitable for the sub-controllers, it should also activate and deactivate the controllers as needed – to respond to the driver's command.

Articulated solutions (solution models):

- **Sol-2. Active control of the shock absorption and chassis clearance** (*new*): See sch. F and all the details in the discussion. (*Transcribed in Appendix B*)

- **Context number 4 (DC-IV)**

Articulated requirements (objectives):

- **Req-1. Control of shock absorption**: The system should be able to control the car suspension/shock absorption.
  - **Req-1.3 Automatically AND manually adjustable absorption**: This is a better alternative, the driver should be able to choose himself if he wants auto-mode or manually sets up the absorber rigidity, toughness.
- **Req-2. Chassis clearance control**: The system should be also able to work with chassis clearance – height above road.
  - **Req-2.3 Automatically and manually adjustable clearance**: Similarly, the driver should be able to say if he wants an automated optimisation of chassis clearance or sets the value manually to address a specific request.

- **Req-3. Need to measure the deflection of the wheel**: In order to control the shock absorbers behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation
- **Req-4. Need to measure also the vibrations**: OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration
- **Req-5. Shock absorber rigidity must be adjustable**: This is essential to have an 'active' absorber, the shock absorbing properties must be adjustable in response to road conditions. Moreover, this should be easy to do!
- **Req-7. Controlling the clearance**: OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.
- **Req-8. Joint controller of absorption AND clearance**: This is a kind of 'master' controller that will coordinate the actions of the two simpler or dedicated sub-controllers. This controller will also be responsible for translating the driver's choices and commands to the values suitable for the sub-controllers, it should also activate and deactivate the controllers as needed – to respond to the driver's command.
- **Req-9. Need to have some 'user interface'** (*new*): This interface is needed to give the driver a friendly control panel and language for expressing his commands. The role of the interface would be to translate qualitative terms such as 'hard', 'soft' or 'high', 'very low' to the quantitative values that can be passed to the appropriate controllers and actuators.

Articulated solutions (solution models):

- **Sol-3. Controlled system with a higher-level controller + interface** (*new*): See the schematic in sketch G.

- **Context number 5 (DC-V)**

Articulated requirements (objectives):

- **Req-3. Need to measure the deflection of the wheel**: In order to control the shock absorbers behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation
- **Req-4. Need to measure also the vibrations**: OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration
- **Req-5. Shock absorber rigidity must be adjustable**: This is essential to have an 'active' absorber, the shock absorbing properties must be adjustable in response to road conditions. Moreover, this should be easy to do!
- **Req-7. Controlling the clearance**: OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.

Articulated solutions (solution models):

- **Sol-4. Details – physical input/output information and control flow** (*new*): See the labelled scheme sketched as H.

T21: Control of paper-smoothing process

Design brief

Design a control device for a paper mill that takes a coil of raw paper as input and ensures that paper at the machine output has desired thickness, is smooth and clean from any debris. Finished product is another coil of a homogenous paper free of any flaws (rippling, tearing, crumpling, ...)

- **Context number 2 (DC-II)**
  Articulated requirements (objectives):
  - **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
  - **Req-2. Ensure that paper is smooth and debris-free**: Another basic req. from the customer: the final product should be polished
  - **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
  - **Req-4. Detect the end of coil**: To avoid paper tearing and/or mill damage, make sure to stop the whole procedure before the actual end of input coil occurs.

  Articulated solutions (solution models):
  - **Sol-1. Sketch of the customer's requirements**: See sketch A in the notebook.

- **Context number 3 (DC-III)**
  Articulated requirements (objectives):
  - **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
  - **Req-2. Ensure that paper is smooth and debris-free**: Another basic req. from the customer: the final product should be polished
  - **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
  - **Req-4. Detect the end of coil**: To avoid paper tearing and/or mill damage, make sure to stop the whole procedure before the actual end of input coil occurs.
  - **Req-5. Dampening of paper before rolling** (*new*): In order to comply with the technology, paper must be dampened before any further manipulation.
  - **Req-6. Rolling mechanism** (*new*): The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.
  - **Req-7. Drying after rolling** (*new*): To store paper, we must ensure it is not damp after rolling = drying mechanism.

  Articulated solutions (solution models):
  - **Sol-2. Paper rolling principle** (*new*): See sketch B showing the principle how to roll paper – simple solution.

- **Context number 4 (DC-IV)**
  Articulated requirements (objectives):
  - **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
  - **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
  - **Req-5. Dampening of paper before rolling** (*new*): In order to comply with the technology, paper must be dampened before any further manipulation.
  - **Req-6. Rolling mechanism**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.
    - **Req-6.1 Extended rolling mechanism** (*new*): Moreover, the absorption must be automatically adjustable in response to the actual terrain, road surface.
  - **Req-7. Drying after rolling**: To store paper, we must ensure it is not damp after rolling = drying mechanism.

  Articulated solutions (solution models):
  - **Sol-3. Principle for rolling and thickness adjustment** (*new*): See sketch C for details – the principle is to re-use the mechanism in a sequence with decreasing gaps between the rolling cylinders.

- **Context number 5 (DC-V)**

  Articulated requirements (objectives):

  - **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
  - **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
  - **Req-5. Dampening of paper before rolling** (*new*): In order to comply with the technology, paper must be dampened before any further manipulation.
  - **Req-6. Rolling mechanism**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.
    - **Req-6.2 Rolling mechanism using alternately placed cylinder pairs** (*new*): The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate – one up, next down, up.
  - **Req-7. Drying after rolling**: To store paper, we must ensure it is not damp after rolling = drying mechanism.

  Articulated solutions (solution models):

  - **Sol-4. Principle for rolling and thickness adjustment – alternate cylinders** (*new*): See sketch C1 for details of this little modification in cylinders layout – the principle is to re-use the mechanism in a sequence with decreasing gaps between the rolling cylinders. We want to 'squeeze' the whole plant to a smaller space...

- **Context number 6 (DC-VI)**

  Articulated requirements (objectives):

  - **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
  - **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
  - **Req-5. Dampening of paper before rolling** (*new*): In order to comply with the technology, paper must be dampened before any further manipulation.
  - **Req-6. Rolling mechanism**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.
    - **Req-6.2 Rolling mechanism using alternately placed cylinder pairs**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate – one up, next down, up.
  - **Req-7. Drying after rolling**: To store paper, we must ensure it is not damp after rolling = drying mechanism.
  - **Req-8. Assume that the thickness of paper is constant** (*new*): We can make this assumption that the thickness of paper on one coil is not changing (for the simplified approach when we adjust the gap once only and keep it as it was set, without the need to adjust).
  - **Req-9. Control according to the input coil** (*new*): Taking previous assumption into account we may say that the parameters influencing the regulation will be given solely by the type and size of input coil.
  - **Req-10. Unrolling of input coil and rolling on the output** (*new*): We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements).
  - **Req-11. Need of a motor to turn the output coil** (*new*): Since we need to move the paper through the machinery, the easiest way is to power the output coil. So we need a motor or something able to turn the coil.

Articulated solutions (solution models):

- **Sol-5. Added control mechanism and paper movement** (*new*): Paper will be moved through the cylinders using the output coil as a pulling and stretching device. For details see sketch D.

- **Context number 7 (DC-VII)**

Articulated requirements (objectives):

- **Req-1. Ensure that paper is ripple-free**: Basic requirement from the customer – final product should be a coil of a smooth paper, without ripples, folds, tears, etc.
- **Req-3. Even thickness of paper**: Another req. directly from the customer – ensure paper has even thickness all along its length
- **Req-5. Dampening of paper before rolling** (*new*): In order to comply with the technology, paper must be dampened before any further manipulation.
- **Req-6. Rolling mechanism**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.
  - **Req-6.2 Rolling mechanism using alternately placed cylinder pairs**: The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate – one up, next down, up.
- **Req-7. Drying after rolling**: To store paper, we must ensure it is not damp after rolling = drying mechanism.
- **Req-8. Assume that the thickness of paper is constant**: We can make this assumption that the thickness of paper on one coil is not changing (for the simplified approach when we adjust the gap once only and keep it as it was set, without the need to adjust).
- **Req-9. Control according to the input coil**: Taking previous assumption into account we may say that the parameters influencing the regulation will be given solely by the type and size of input coil.
- **Req-10. Unrolling of input coil and rolling on the output**: We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements).
  - **Req-10.1 Unrolling of input coil and rolling on the output** (*new*): We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements). We can achieve that quite easily when we change 'unrolling' and 'rolling' (thus refinining the paper flow): to pull paper off the input coil, to pull it towards the output through the whole machinery, to coil it tightly at the output.
- **Req-11. Need of a motor to turn the output coil**: Since we need to move the paper through the machinery, the easiest way is to power the output coil. So we need a motor or something able to turn the coil.
- **Req-12. Need of a motor for input coil** (*new*): This is the result of solving a potential problem with tearing wet paper, we need something to pull the paper off the input coil, something that is not located on the coil itself (no pushing) but perhaps a little bit after it so that it can actually pull paper very gently but rather efficiently.

Articulated solutions (solution models):

- **Sol-6. Added control mechanism and paper movement** (*new*): Paper will be moved through the cylinders using the output coil as a pulling and stretching device, and simultaneously there will be another pulling device closer to the input coil to ensure even unrolling and prevent paper tearing. For details see sketch D1.

Appendix B: Design decisions justification (transcription)

T11: Control of active shock absorption

- **J-489. Design brief**: Please, specify here any changes, amendments, suggestions to the initial design brief.
  - **J-490. Specification**: (1) Automated adjustment of shock absorption (terrain), (2) Manual adjustment of chassis clearance (for different terrain), and (3) Automated adjustment of clearance
  - **J-495. Active suspension = ?**: What we expect from an active shock absorber
    - **J-496. Ideal state**: Ideally, the driver should not feel any bumps or changes of the road surface. The system should intelligently absorb and eliminate the uneven road.
    - **J-497. Specification, principle**: We can take passive absorber and add some logic to adjust its rigidity according to road surface. For smooth road we can have a tougher absorber (sporty); for bumps we need softer one.
      - **J-498. Required components**: To control shock absorber we need a sensor for road 'unevenness', some processing logic for adjusting the absorber, and the actual adjustable absorber.
    - **J-499. Manual inputs**: Active suspension needs some input so that driver can set what he wants. The manual input will override the adjusted value for the absorption.
  - **J-500. Automated clearance**: This should behave so that when the load in a car changes, the clearance is adjusted too. The adjustment should be automated – always or once?
    - **J-501. Principles**: We can measure the actual clearance, logic would evaluate and some actuator would make the changes, to get into initial or recommended height.
    - **J-502. Manual input**: Manual input is a request from the driver to change clearance immediately; e.g., change of terrain, off-road ride. Manual input gives the new value to maintain the clearance on.

- **J-491. Chassis clearance = const.**: Constant clearance simpler case – assume we work with a given value, first.
  - **J-492. Suspension = const.**: (1) If nothing changes in the absorber =¿ plain, passive device
    - **J-494. Nothing new**: This is useless to continue, we can forget any passive device.
  - **J-493. Suspension − variable**: Active absorber should be adjustable
    - **J-503. How it works**: When wheel gets on a bump, its position against the chassis changes, sketch B. Spring inside the absorber should minimise the transfer of deflection on the chassis. Bigger bump = bigger deflection, hence, faster elimination is needed.
      - **J-508. Just not spring...**: The problem is that the whole thing cannot be realised as a spring – it would rather tricky to adjust the toughness of springs; need something that can be changed easier (hydro, pneu?)

- **J-509. Pneumatic piston**: OK,take a pneumatic piston instead of spring – the same behaviour, only rigidity is given by the volume/pressure of air inside the piston. That would need a simple pump to increase pressure and vent to release air (= softer abosrber)

J-504. Sensor + details: Sensor measures delta between wheel axle and chassis. This can be a device measuring distance wheel-chassis placed on the absorber. Measured value goes to logic – control alg.

J-505. Control algorithm: This is the 'logic' that takes deflection of a wheel from normal position, and decides how to change the current toughness of the absorber.

- **J-506. Source of deflection/vibration (too hard abosrbers)**: I. With too hard springs we enter bumpy road, immediately transfered onto chassis = car vibrates = make absorber softer.
- **J-507. Source of vibration – too soft abs.**: II. Too soft suspension on smooth surface may also trigger vibration. In this case, we need to make it harder, so that the car is less sensitive.
- **J-506. Source of deflection/vibration (too hard abosrbers)**: The transition from smooth to bumpy road will first affect the wheel and a second later chassis, i.e., deflect BEFORE chassis. When we go from bumpy onto a smooth road, the deflection is minimal/none but vibration still lasts, and it may even force the wheel to deflect unnecessarily, i.e., vibration and NO deflection (or better deflection AFTER vibration)
- **J-510. Minimise deflection – not good**: Measuring the position of wheel to the chassis is not good, because the ideal state is 'no deflection' = rigid rod... Of course, then we don't need any absorbers when we start with rods; the bump has bad effect on chassis. So, this will need to be checked by the control logic – a max. limit on pressure, perhaps.
- **J-511. Need to measure vibration on chassis**: The wheel deflection can remain – simple measurement, but we need another sensor measuring the magnitude of the chassis vibration. In this way, we have two sources from measurement, they can validate each other.
- **J-513. Need to trace the order between deflection measurements**: So the rule can be
  IF deflection BEFORE vibration THEN decrease pressure
  IF deflection AFTER vibration THEN increase pressure
  (increase pressure = pump more air into piston (see sketch C))

J-514. SOL1: Controlling active suspension: See sch. D., the chassis clearance is const. not included in the control alg./schema.

J-515. The rules extended: The control logic has two simple rules that take as input the values measured on absorber (deflect/vibrate). one more thing that the controller will have to take care is that 'rigid rod' problem if trying to increase pressure too much, and the other thing is certain insensitivity in the rule firing.

- **J-516. 'Try & see' heuristic**: This could be a heuristic rule for the control of suspension incorporating the insensitivity region, so that the controller does not switch the pump on/off too often. When nothing's happening, we try to increase pressure a little, and see what effect that has on the deflection/vibration. If there is no effect or a positive effect (the vibration disappears) then continue with this heuristic. End it when you reach a kind of 'saturation' = it starts to worsen the vibraitons. The diff betwen the original pressure given by rule and this heuristic one, forms a region of absorber insensitivity. This heuristic may be perhaps applied reversely, i.e. try to go softer, and see what happens. This could help on a terrain that becomes uneven 'continuously'.

J-517. Make it harder or softer first?: The question is what shall be done first in a general case - try to increase the pressure or decrease it? This can be helped again by the order in which the disturbances are captured by sensor. So, if car enters a bumpy terrain, the wheel deflection shall be captured before the chassis vibration.
- In such case does not make sense to increase pressure any further, on the contrary. When car enters a smooth road from an off-road terrain, the chassis may continue to vibrate, so the pressure may be increased. We may perhaps measure some kind of dependency of the delay in the measurements, and use this as an estimate of what may happen in the next moments - prediction?!?

J-518. Clearance-variable, suspension-variable: So far, we worked with constant and unchanging clearance of chassis from the ground; only the suspension was adjustable. We can try to take also the effect of chaning load distribution on the control. The clearance adjustment can be done mechanically, kind of screw going up and down – see illustr. E.

J-519. Relation clearance – suspension: In principle, chassis clearance may be independent from the suspension – if there is certain minimal height maintained and if the load does not exceed some maximum. But suspension may be influenced by uneven distribution of load on board - the load may shift, e.g in bends or uphill.

J-520. Definition of adjustable clearance: Clearance may be adjusted manually (command from driver) or automatically. In both cases, the adjustments can be done without caring for load distribution, and the right values could be tuned later – should be easier to tune than make big changes. In manual mode, the driver sets the requested clearance, the system changes it and maintains the car on this value (if poss.) In auto-mode, depending on the type of terrain (given by sensors used for suspension - vibration), we can set also the clerance.

J-522. Manual mode principle: Driver enters the value before starting the ride, this must be maintained during the ride.

J-523. Adjust when stationary: We may check the values of piston compression and screw extrusion before starting the ride. From these values we compute the current clearance. Now we can adjust screw up/down as needed to get requested clearance.

J-524. Stationary vehicle?: Classical physics – vehicle behaves similarly when stationary and moving steadily with constant velocity. So shuld we allow to change the clerance when moving steadily? Then, when we have load aboard and it's loosely fitted, it may shift around. Change in the load distribution affects the wheels differently. Seems that we will need to adjust the clearance also while driving = more difficult, less safe?

J-525. Median of piston position: Instead measuring piston/screw extrusion when stationary, we may use a median value for each piston as measured during a ride. In this way, we can adjust value of clearance also while driving. In certain small limits, this should be pretty safe.

J-526. Automated mode: This is an adjustment automatically responding to the terrain type – with changed suspension, a similar set of rules adjusts also clearance.

J-527. Translation table: We can model how different terrains affect the effective behaviour of the piston, and how clearance influences this behaviour. From these findings we may construct a table that translates the values of sensors into 'terrain types'. Each terrain type may be also assigned an optimal clearance; thus auto-mode would only grab the values from sensors, finds an appropriate value in the table, and send it to the actuator/motor.

J-521. Implementation: The clearance can be changed mechanically – using a screw connected to the absorbing piston and moving into chassis. it can be controlled by a small motor commonly available. Since, the wheel dimensions and screw pitch is known, it is easy to calculate how much the screw needs to be turned to effect particular clearance.

- **J-528. Chassis vibration sensor**: OK,let's have a look at how can we measure the vibration of the chassis – this is crucial for the control logic.
  - **J-529. Sensor from airbags**: To measure the actual value of chassis vibration we can re-use the devices commonly appearing in the electronics for airbags. This is basically the same need, find out when chassis vibrates faster than it would naturally = we found a bump = control action.

- **J-530. Piston deflection sensor**: This is a complementary measurement on the absorber that gives the actual position of a wheel agains chassis base.
  - **J-531. Deflection = diff distance**: To measure deflection we may use a simple distance sensor, e.g. resistance-based measurement is pretty robust and precise. Also, we may a trivial incremental impulse generator or anything similar.

- **J-532. Adjustment of absorber**: To have an active suspension, we must be able to adjust the absorber's properties (rigidity or toughness) – easily!!
  - **J-533. Adjustment of pressure inside piston**: Whether we use pneumatic or hydraulic pistons, they both give the same fuctionality; they both contain media that can be compressed – perhaps air is easier compressible than oil or other liquids = so, change in air pressure with a constant volume affects the rigidity of the whole pneu absorber

- **J-534. Suspension control logic**: To implement the control, we have to interpret the verbal rules in a suitable machine language
  - **J-535. Hardware leaving aside**: The IF-THEN rules and heuristics seem to be pretty straightforward to translate to logical gates or perhaps, we can use a car computer, modern cars have it anyway, so we can use it for the active suspension as well.

- **J-536. Clearance adjustment**: As we decided, this should be a simple, robust and safe mechanism that can withstand heavy loads and performs safely in higher speeds.
  - **J-537. Jack/screw with an incremental motor**: The incremental motor is a good device because we won't need another sensor measuring the extrusion – we need only remember the current values and changes in the increments.

- **J-538. SOL2: Control of suspension and clearance (together)**: This is basically an extension of the previous control loop, see sch. F.
  - **J-539. Co-ordinating controller**: Seems that we need a master controller that would coordinate the work of the specialised controllers – one specialises on suspension, the other on clearance.
    - **J-540. Control inputs for suspension ctrl**: On of the roles of 'master' would be translate the driver's commands into values recognisable by the sub-controllers. A kind of switch/input is needed to switch between manual and auto modes. Depending on this switch, the suspension controller can be overriden or left working. Also, this master would translate a type of terrain into a value for the clerance controller, activate the heuristic rules, etc.
    - **J-541. Control inputs for clearance adj.**: Similarly as with suspension, if the switch is in auto mode, then the sub-controller has full responsibility for determining the optimal clearance (tables, heuristics). Otherwise, in the manual mode, the driver specifies the requested value and if this can be achieved, the controller will maintain it – regardless of optimality; the rules defined earlier still apply, only using a master controller, we may activate or deactivate any of them (as needed)

- **J-543. Interfaces**: This is an extension if we want to support a comfortable communication of the driver's requests/commands to the controllers.
  - **J-544. Information channel**: OK, this would be mainly a translator changing the intuitive values from the driver (e.g. from a dial setting softer/harder suspension) to the specific quantitative values for controllers (e.g. X kPa for pressure or Y cm for clearance).
    - **J-545. Suspension controls**: This can be somewhat similar to the knob for A/C control in cars - say Position 0 = automated mode, Position 1 = very soft, Position 2 = soft, etc. (like a heater). Physically, we need two channels, first saying auto/manual (on/off or 1/0). Second would be carrying the actual signal corresponding to the qualitative adjustment (e.g. 0.25, 1.25, 1.75, ...)

- **J-546. Clearance controls**: Again the same principle – first, on/off = auto or manual. Another wire would transfer the coded value for a particular cleraance adjsutment (e.g. high = 40cm, medium = 30 cm, low = 20cm,...) Interface can be again constructed as a simple dial, it may be also digital, but that seems to be too complex to be worthwhile... but it's possible.

- **J-547. SOL3: Control system with user interface**: Sch. F remains, only an interface is added, as above.

- **J-548. SOL4: Physical details and input/output flow**: See control the labelled sketch H (hope it's not too messy..)

T21: Control of paper-smoothing process

- **J-690. Design brief**: Please, specify here any changes, amendments, suggestions to the initial design brief.
  - **J-692. Technology sketch**: As an input we take a coil of (possibly) rippled, raw paper that should be transformed into a coil of paper with given requirements. For details, sketch A
  - **J-693. Desired output**: What are desired parameters at the output?
    - **J-694. Re: Desired output**: Paper should have no ripples, be polished and of even thickness. Can I assume the thickness will not change significantly?
      - **J-695. Re: Max. thickness reduction**: Yes, basically the maximal reduction should not exceed 15%, polishing and stretching are more important in this part of process.
  - **J-696. Other restrictions from Customer**: Are there any other related restrictions - either from Customer or technology?
    - **J-697. Re: Other restrictions from Customer**: Perhaps, what about the weight and diameter of the coils?
      - **J-699. Probably irrelevant**: Seems to be irrelevant because the mass of paper cannot change during the process. Also, the diameter of the output coil will be at most equal to the input one - paper will be smooth = less space
  - **J-700. Start and end of the process**: How does the process begin and how does it end? Any possible pitfalls there?
    - **J-703. Re: Start and end of the process**: Before the machine is launched, the coil at the input, a small portion is unrolled and manually fed to the output coil. The process should be finished before the paper is torn off the input!
      - **J-704. Checking the end of coil**: ... The process should be finished before the paper is torn off the input coil! – Then I would suppose this is another requirement on the control device

---

- **J-707. SOL1: Initial sketch**: See sketch A in my notebook
  - **J-708. Re: SOL1: Initial sketch**: Here, we have the desired function of the plant – now we can look at the details, what is actually involved.
    - **J-709. Paper rolling**: Basic functionality is rolling, according to the technology, damped paper is better to shape, however we must ensure the 'right amount' so that it's not too wet.
      - **J-711. Pre- and post-adjustment**: Before the actual rolling mechanism, we need some moisturiser... and again, the paper on output should be dry – after rolling we need a drying tunnel or something like that.

- **J-717. Smooth paper (damping, rolling, drying)**: To have paper even and smooth I suggest the technology of rolling that should remove all ripples, etc.
  - **J-718. Paper damping**: Before any manipulation with paper it should dampened, afterwards it may be shaped.

- **J-719. Paper rolling and smoothing**: To achieve even paper we use a principle of rolling using a simple pair of two cylinders.
  - **J-720. Paper drying and storing on a coil**: And finally, the reverse operation to dampening should be drying to comply with technology.

- **J-721. SOL2: Paper smoothing**: Taking into account all these basic requirements, we should take the simplest solution, as suggested by sketch B.
  - **J-722. Re: SOL2: Paper smoothing**: Works fine for the current requirements.

- **J-723. Extended rolling (incl. pressing)**: Now, we may attend the issue of reducing the thickness. As we learned in the disc. above, the thickness is not being changed significantly.
  - **J-724. Dampening**: Keep the dampening as in the simple solution.
  - **J-725. Mechanism for pressing**: Basically, the previous mechanism of smoothing works fine and may be reused for reducing the thickness as well – to avoid another dampening.
    - **J-726. Sequence of pressing cylinders**: After rolling and smoothing but before drying we can introduce two-three pairs of similar cylinders. The sequence of cylinders can reduce the thickness.
      - **J-727. Re: Sequence of pressing cylinders**: This thickness reducing cylinders is in principle not different from the smoothing – they may be combined together.
      - **J-738. Sequence of alternate cylinders**: To fix a minor flaw of the straight sequence of cylinders, we require their alternate layout.

- **J-728. SOL3: Smoothing and thickness**: See sketch C for details of the changed layout.
  - **J-731. Not sure of quality**: A little problem with this arrangement is that it may be rather long, which might be inpractical for maintenance and control. Also there may arise a danger of pressing debris into paper instead of removing them.
    - **J-733. Squeezing cylinders**: We can 'squeeze' cylinders closer but again, there must be a little gap between them to ensure smooth operation. And probably that may not help much.
      - **J-734. Re: Squeezing cylinders**: Well, and what about 'squeezing' them so they shift a bit up and down – they don't have to be in a row, they may be one up, one down ... and paper may run through them.
      - **J-736. Alternate layout of cylinders**: Sounds good and doesn't require a major change in the principles, just a little reshuffling of parts.
      - **J-742. SOL4: Modification with using alternate cyl.**: See sketch C1 for details including that idea of 'squeezing' cylinders together.

- **J-754. Process of regulation and control**: Now we can look at the regulation itself
  - **J-755. Assumption of constant thickness on output**: To start with we assume that the desired thickness of paper is not changing during the milling and is given at the beginning by the type of the input coil.
    - **J-756. Regulation based on input coil**: Following this assumption we may design a simple direct control based on the parameters detected for the input coil.
  - **J-759. Paper flow through cylinders**: The controlled device will be the output coil that must ensure both, the pulling of paper from input through the machinery of cylinders and rolling it smoothly on the output. This is the simplest case ...
    - **J-762. Re: Paper flow through cylinders**: Attach the motor to the output coil and power it sufficiently to mvoe the paper and roll it tightly.

- **J-765. SOL5: Regulation and drive**: As sketched in Fig.D
  - **J-766. Danger of paper tearing**: This solution satisfies the requirements however, paper when damp is a bit vulnerable concerning tearing – esp. if it's only pulled through a rather complex maze of cylinders.
    - **J-767. Both coils driven**: A solution to prevent paper damge would be perhaps powering also the input coil, thus having drive on two different locations and spread the tension more evenly.

J-768. Re: Both coils driven: But when the output is pulling, the input should
- be 'pushing' which may be tricky and may create more folds and ripples (paper is not sturdy enough)...

J-769. Simple pulling device attached to input coil: We may try another alternative, where the paper is actually being pulled by a motor directly at the input, then
- fed in to a cylinder maze and pulled again at the end of the plant (output coil). This ensures that paper is actually pulled out when it is still dry, so the danger of tearing is reduced – we actually perform the task before the problem may occur.

J-770. SOL6: Paper moving on two places: Sketch D1 shows a principle of
- pulling paper closer to the input thus reducing the danger of tearing.

This page intentionally left blank